ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Τμήμα Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων

Διπλωματική Εργασία

Θέμα:

" Ανάπτυξη αλγορίθμου κωδικοποίησης δικτύου για δίκτυα ομότιμων κόμβων "

Παναγόπουλος Πέτρος

Επιβλέποντες:

Αντώνης Αργυρίου

Λέανδρος Τασιούλας

- Βόλος 2011 -

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τον κ. Λέανδρο Τασιούλα, καθώς και τον κ. Αντώνη Αργυρίου η συμβολή του οποίου ήταν ιδιαίτερα σημαντική για την εξέλιξη και την ολοκλήρωση της διπλωματικής εργασίας.

Περιεχόμενα

Εισαγωγή

Ένα δίκτυο ομότιμων κόμβων ή διαφορετικά ένα δίκτυο peer-to-peer (ομότιμος προς ομότιμο), είναι ένα δίκτυο το οποίο αποτελείται από δύο ή περισσότερους συνδεδεμένους υπολογιστές με ίσα διακαιώματα ο ένας προς τον άλλον, με τη δυνατότητα να διαμοιράζονται τους πόρους τους. Πόροι όπως είναι η επεξεργαστική ισχύ του κάθε κόμβου, το εύρος ζώνης αλλά και ο χώρος αποθήκευσης. Τις περισσότερες φορές, ένα δίκτυο ομότιμων κόμβων αποτελεί ένα επικαλύπτον δίκτυο (overlay network) πάνω από το ήδη υπάρχον διαδίκτυο, όπως το διαδίκτυο είναι ένα επικαλύπτον δίκτυο πάνω από το τηλεφωνικό δίκτυο. Σε ένα επικαλύπτον δίκτυο ομότιμων κόμβων τα τερματικά είναι εικονικά απευθείας συνδεδεμένα μεταξύ τους με τη δυνατότητα επικοινωνίας.

Αυτό που χρειάζεται κάθε υπολογιστής για να συνδεθεί και να επικοινωνήσει με έναν άλλο υπολογιστή του ομότιμου δικτύου, είναι η εγκατάσταση μιας εφαρμογής, η οποία να υλοποιεί ένα συγκεκριμένο πρωτόκολλο επικοινωνίας. Το πρωτόκολλο αυτό θα είναι ίδιο για όλους τους κόμβους του δικτύου, έτσι ώστε να επικοινωνούν επιτυχώς μεταξύ τους. Από τις πιο επικρατούσες λειτουργίες σε δίκτυα ομότιμων κόμβων είναι ο διαμοιρασμός ενός αρχείου ανάμεσα τους. Από τα πιο διαδεδομένα πρωτόκολλα για διαμοιρασμό αρχείου σε peer-to-peer δίκτυα, αποτελεί το BitTorrent πρωτόκολλο. Όπως κάθε άλλο ευρέως χρησιμοποιούμενο πρωτόκολλο επικοινωνίας μεταξύ κόμβων, έτσι και αυτό βασίζεται σε βασικές αρχές επικοινωνίας και ανταλλαγής δεδομένων.

Η κωδικοποίηση δικτύου (network coding) αποτελεί ένα πρόσφατο σχετικά πεδίο στη θεωρία πληροφορίας, η οποία αποκλίνει από τη βασική αρχή ροής πληροφορίας σε ένα δίκτυο. Η κωδικοποίηση δικτύου επιτρέπει και ενθαρρύνει την μίξη της πληροφορίας πριν την αποστολή της σε κάποιον άλλο κόμβο, με στόχο τη βελτίωση του throughput, δηλαδή τη λήψη περισσότερων δεδομένων σε μονάδα χρόνου, σε σύγκριση με τον παραδοσιακό τρόπο μετάδοσης της πληροφορίας. Πάνω στην τεχνική της κωδικοποίησης δικτύου βασίζεται το Avalanche, ένα νέο σύστημα διαμοιρασμού αρχείου με ανταλλαγή κωδικοποιημένης πληροφορίας σε δίκτυα ομότιμων κόμβων, το οποίο υπόσχεται αρκετές βελτιώσεις σε θέματα επίδοσης συγκριτικά με το επικρατέστερο έως τώρα BitTorrent.

Στην παρούσα εργασία μελετάμε ορισμένες πτυχές της χρήσης της τεχνικής κωδικοποίησης δικτύου σε δίκτυα ομότιμων κόμβων. Για τον σκοπό αυτό, σχεδιάσαμε και υλοποιήσαμε ένα υβριδικό σύστημα ομότιμων κόμβων, το οποίο βασίζεται εν μέρει στις τεχνικές του BitTorent και του Avalanche. Η βασική ιδέα του προτεινόμενου συστήματος έγκειται στο ότι ένας κόμβος επιλέγει την μετάδοση η μη κωδικοποιημένων πακέτων, βάση μίας μετρικής, αλλά και της πληροφορίας σχετικά με τα διαθέσιμα πακέτα στους γειτονικούς κόμβους. Το σύστημα αυτό προσομοιώθηκε για μια πληθώρα διαφορετικών παραμέτρων.

Το σύστημα BitTorrent

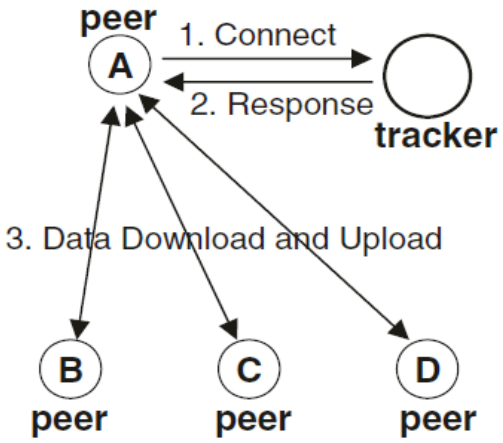
Το BitTorrent αποτελεί ένα πρωτόκολλο διαμοιρασμού αρχείων μεγάλου μεγέθους, βασισμένο στην αρχιτεκτονική peer to peer. Αναπτύχθηκε το 2002 από τον Bram Cohen και γνώρισε μεγάλη ανταπόκριση από το διαδικτυακό κοινό, κάτι το οποίο συνεχίζεται έως και σήμερα. Το BitTorrent επιτρέπει τη συνεργασία μεταξύ κόμβων του δικτύου, χωρίζοντας το αρχικό αρχείο σε μικρότερα κομμάτια μεγέθους 256KBytes, το οποίο επιτρέπει στους κόμβους να κατεβάζουν κομμάτια παράλληλα, από διαφορετικούς κόμβους. Η λήψη του αρχείου θα έχει ολοκληρωθεί από έναν χρήστη, όταν αυτός θα έχει παραλάβει όλα τα μικρότερα κομμάτια.

Πιο αναλυτικά, το BitTorrent πρωτόκολλο αξιοποιείται από μεμονωμένους χρήστες για το διαμοιρασμό ενός αρχείου μεταξύ τους. Το σύνολο των χρηστών που μετέχουν στο διαμοιρασμό ονομάζεται peers. Οι peers υποδιαιρούνται σε δύο κατηγορίες, τους seeders (complete downloaders) και τους leechers (incomplete downloaders). Οι seeders είναι αυτοί οι οποίοι κατέχουν ολοκληρωμένο το αρχείο και ο ρόλος τους στο p2p δίκτυο είναι να στέλνουν κομμάτια του αρχείου σε άλλους κόμβους, οι οποίοι επιθυμούν να το ολοκληρώσουν. Αντίστοιχα, οι leechers είναι αυτοί οι οποίοι δέχονται κομμάτια του αρχείου από τους υπόλοιπους κόμβους στο p2p δίκτυο, με στόχο να ολοκληρώσουν το αρχείο. Παράλληλα, κάθε leecher αναλαμβάνει και το ρόλο του seeder, στέλνοντας πακέτα σε άλλους κόμβους οι οποίοι επιθυμούν να ολοκληρώσουν το αρχείο. Σύμφωνα με το πρωτόκολλο, κάθε κόμβος ο οποίος ολοκληρώνει ένα κομμάτι του αρχείου, μπορεί να συνεισφέρει στο διαμοιρασμό του κομματιού αυτού. Η επιλογή του κομματιού από τους κόμβους, βασίζεται σε συγκεκριμένες πολιτικές, οι οποίες αναφέρονται παρακάτω.

Για να ξεκινήσει και να επιτευχθεί όλη αυτή η διαδικασία που υποστηρίζεται από το BitTorrent, είναι απαραίτητη η ύπαρξη ενός αρχείου με κατάληξη .torrent στους χρήστες. Το .torrent περιέχει πληροφορίες για το επιθυμητό αρχείο, όπως το μέγεθός του, το όνομα του και τη διεύθυνση του tracker. Οι trackers είναι υπεύθυνοι στο να βοηθήσουν τους peers να συνδεθούν κατάλληλα μεταξύ τους. Ο χρήστης κατεβάζει το .torrent και το ανοίγει με μια εφαρμογή που υλοποιεί το πρωτόκολλο. Ο tracker μέσω της εφαρμογής ενημερώνεται για την ύπαρξη ενός νέου χρήστη. Ο χρήστης ανταλλάσσει πληροφορίες με τον tracker χρήσιμες για την έναρξη της διαδικασίας, όπως για το ποιο αρχείο κατεβάζει, σε ποια πόρτα ακούει και ο tracker του απαντά με μία λίστα κόμβων οι οποίοι διαμοιράζονται το ίδιο αρχείο. Η επικοινωνία με τον tracker γίνεται με HTTP αιτήσεις πάνω από TCP αλλά υποστηρίζεται και επικοινωνία με UDP συνδέσεις. Εν συνεχεία, ο κόμβος που επικοινώνησε με τον tracker, συνδέεται με τους κόμβους που του υπέδειξε σχηματίζοντας έτσι μια γειτονιά κόμβων και η διαδικασία διαμοιρασμού και απόκτησης του αρχείου ξεκινάει. Η επικοινωνία με τον tracker συμβαίνει και μετά την έναρξη του διαμοιρασμού, είτε για να συνδεθούν με άλλους κόμβους, είτε για να προσκομίσουν στατιστικά στοιχεία στον tracker.

Η επικοινωνία με τον tracker με UDP έχει στόχο να μειώσει σημαντικά την πλεονάζουσα κίνηση στον tracker. Για παράδειγμα, χρησιμοποιώντας HTTP, εισάγουμε στο σύστημα σημαντικό πλεόνασμα. Πλεόνασμα στο ethernet επίπεδο (14 bytes ανά πακέτο), στο IP επίπεδο (20 bytes ανά πακέτο), στο TCP επίπεδο (20 bytes ανά πακέτο) και στο

HTTP επίπεδο. Περίπου δέκα πακέτα χρησιμοποιούνται για αίτηση – απάντηση, μέσα σε ένα δίκτυο 50 ομότιμων κόμβων, ο συνολικός αριθμός bytes είναι περίπου 1206 Bytes [1]. Αυτό το πλεόνασμα μπορεί να μειωθεί σημαντικά χρησιμοποιώντας πρωτόκολλο βασισμένο σε UDP, το οποίο χρησιμοποιεί 4 πακέτα και περίπου 618bytes μειώνοντας την κίνηση κατά 50%. Για τον tracker αυτό είναι αρκετά σημαντικό αφού έχει να εξυπηρετήσει χιλιάδες peers,αυξάνοντας έτσι την απόδοσή του.



Σχήμα 1: Επικοινωνία Peer - Tracker και σύνδεση με τους υπόλοιπους peers.

Ας δούμε τώρα κάποια πιο εξειδικευμένα χαρακτηριστικά του πρωτοκόλλου. Ένα ζήτημα για τους κόμβους είναι η επιλογή του κομματιού για downloading. Γενικά, σε αυτό το μέρος η πολιτική που ακολουθεί το πρωτόκολλο, είναι η επιλογή κομματιού που είναι το πιο σπάνιο ανάμεσα στους κόμβους (rarest first policy). Όταν ένας κόμβος ξεκινά το downloading, είναι προφανές ότι δεν έχει κάποιο κομμάτι για να κατεβάσουν από αυτόν. Έτσι, είναι σημαντικό να αποκτήσει ένα κομμάτι όσο το δυνατόν γρηγορότερα. Στην περίπτωση που επέλεγε να κατεβάσει ένα σπάνιο κομμάτι, πολύ πιθανόν να αργούσε να το αποκτήσει, καθώς λίγοι θα είναι οι κόμβοι οι οποίοι θα μπορούν να του στείλουν δεδομένα. Ακολουθεί σε αρχικό στάδιο την πολιτική της τυχαίας επιλογής κομματιού (random first piece) και όταν ολοκληρωθεί η απόκτηση του, η πολιτική αλλάζει σε rarest first piece. Η πολιτική αυτή εξασφαλίζει ότι οι κόμβοι θα έχουν πακέτα που θα ζητούν να αποκτήσουν οι υπόλοιποι κόμβοι, καθώς επίσης και τον ομαλό διαμοιρασμό προσπαθώντας να επιτύχει ομοιόμορφη κατανομή των κομματιών ανάμεσα στους κόμβους.

Επίσης, το πρωτόκολλο υλοποιεί κάποιους μηχανισμούς, ώστε να εξασφαλίζεται η δίκαιη συμπεριφορά των κόμβων απέναντι στους υπόλοιπους. Κάθε κόμβος είναι υπεύθυνος για τη μεγιστοποίηση του δικού του ρυθμού downloading. Για να το πετύχουν αυτό, οι κόμβοι κατεβάζουν από όποιον κόμβο μπορούν και αποφασίζουν σε ποιους κόμβους θα κάνουν upload χρησιμοποιώντας τη συνεργατική στρατηγική tit for tat. Έτσι, κάποιος κόμβος επιλέγει αν θα συνεργαστεί με κάποιον άλλο, άρα θα του κάνει upload

δεδομένα ή αν δε θα συνεργαστεί και δε θα του κάνει upload (choke the peer) τουλάχιστον για ένα προσωρινό διάστημα. Πέρα της δίκαιης συμπεριφοράς των κόμβων, το ζήτημα είναι και τεχνικό αφού το πρωτόκολλο TCP, το οποίο χρησιμοποιείται για την ανταλλαγή των δεδομένων, δεν αποδίδει καλά όταν υπάρχουν αρκετές συνδέσεις.

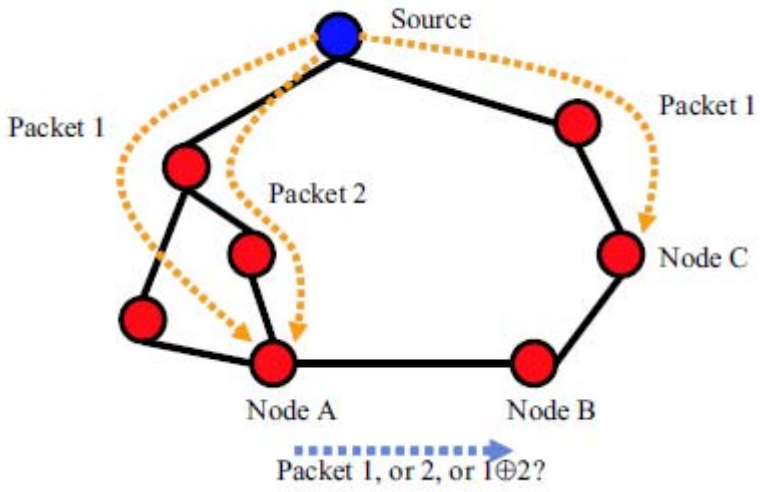
Ένα ακόμα χαρακτηριστικό του BitTorrent πρωτοκόλλου είναι ότι χρησιμοποιεί έναν Distributed Hash Table αλγόριθμο, ο οποίος βασίζεται στο σύστημα Kademlia, για να αποθηκεύει πληροφορίες εύρεσης κόμβων για trackerless .torrents αρχεία [2]. Στην ουσία κάθε peer λειτουργεί και ως tracker. Γενικότερα, σε ένα DHT σύστημα υπάρχουν κόμβοι και κλειδιά-keys. Τα κλειδιά παράγονται μέσω συναρτήσεων κατακερματισμού και αντιστοιχούν σε ένα αρχείο. Κάθε κόμβος αντιστοιχεί σε ένα μοναδικό id και έχει στην κατοχή του κάποιο εύρος κλειδιών. Επίσης, κάθε κόμβος έχει τον δικό του πίνακα δρομολόγησης στον οποίο αποθηκεύει αναγνωριστικά γειτονικών κόμβων και κλειδιά. Η βασικότερη λειτουργία από έναν κόμβο σε ένα DHT σύστημα, είναι η προσπάθεια εύρεσης ενός κλειδιού, lookup(key), όπου ο κόμβος μαθαίνει το αναγνωριστικό του άλλου κόμβου που κατέχει το ζητούμενο κλειδί. Ύστερα, μπορεί να συνδεθεί με τον κόμβο και να κατεβάσει το ζητούμενο αρχείο.

Το σύστημα Avalanche

Το Avalanche είναι ένα νέο σύστημα διαμοιρασμού μεγάλων αρχείων σε δίκτυα ομότιμων κόμβων, το οποίο παρουσιάστηκε το 2005, από τους Christos Gkantsidis και Pablo Rodriguez, βασιζόμενο στην τεχνική κωδικοποίησης δικτύου - network coding. Είναι η πρώτη υλοποίηση με χρήση αυτής της τεχνικής σε δίκτυα ομότιμων κόμβων και σύμφωνα με τα όσα έχουν ήδη δημοσιευτεί και εξεταστεί με προσομοιώσεις [3] και με πειράματα σε πραγματικό περιβάλλον [4], διαπιστώνεται ότι υπάρχει όφελος 2-3 φορές, σχετικά με τον χρόνο απόκτησης του αρχείου, σε σύγκριση με τον διαμοιρασμό της πληροφορίας με μη κωδικοποιημένο τρόπο.

Ένα από τα θέματα που ξεπερνάει το Avalanche σε σχέση με το BitTorrent είναι αυτό του κατάλληλου προγραμματισμού διαμοιρασμού της πληροφορίας στο δίκτυο των κόμβων. Όπως αναφέραμε προηγούμενα το BitTorrent χρησιμοποιεί αλγορίθμους DHT για να βρεθεί κάποιο κομμάτι που χρειάζεται ένας κόμβος τη δεδομένη στιγμή. Στο Avalanche αυτό που γίνεται είναι κάθε κόμβος να συνδυάζει ότι κομμάτι του αρχείου έχει και να το στέλνει στον ενδιαφερόμενο κόμβο.

Για να γίνει πιο κατανοητή η λειτουργία του συστήματος θα χρησιμοποιήσουμε ένα παράδειγμα. Ας υποθέσουμε ότι έχουμε την τοπολογία η οποία φαίνεται στο Σχήμα 2. Ο κόμβος A λαμβάνει από την πηγή τα πακέτα 1, 2 και ο κόμβος C λαμβάνει το πακέτο 1. Ανάμεσα στον A και στον C όπως βλέπουμε παρεμβάλλεται ο B. Αν ο B κατεβάσει από τον A το πακέτο 1, τότε ο σύνδεσμος ανάμεσα στον B και στον C θα είναι άχρηστος, αφού και οι δύο κόμβοι έχουν τα ίδια πακέτα. Ο B πρέπει να ξέρει ότι ο C έχει το πακέτο 1, έτσι ώστε να επιλέξει να κατεβάσει το πακέτο 2 από τον A και ύστερα να μπορεί να χρησιμοποιηθεί ο σύνδεσμος ανάμεσα στον B και C. Με χρήση όμως της κωδικοποίησης, ο κόμβος B μπορεί να κατεβάσει από τον A, το συνδυασμό των πακέτων 1, 2 και ύστερα να μπορεί να χρησιμοποιηθεί ο σύνδεσμος μεταξύ B και C. Ο B θα μπορεί να στείλει στον C τον κωδικοποιημένο συνδυασμό των πακέτων 1, 2 και ο C αφού έχει ήδη το πακέτο 1 να μπορεί να αποκωδικοποιήσει το μήνυμα και να ανακτήσει το πακέτο 2.

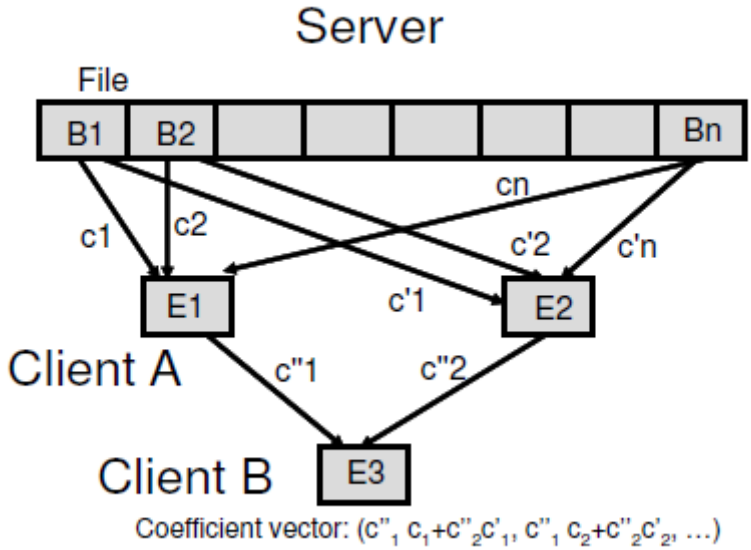


Σχήμα 2: Τοπολογία δικτύου στην οποία διακρίνεται όφελος από την κωδικοποίηση δικτύου.

Όπως σε κάθε σύστημα ομότιμων κόμβων όπου οι κόμβοι θέλουν να συνεργαστούν για να διαμοιραστούν ένα αρχείο, έτσι και σε αυτή την περίπτωση υπάρχει ένα πλήθος κόμβων οι οποίοι επιθυμούν να συνεργαστούν ώστε να αποκτήσουν ένα αρχείο. Σύμφωνα με το Avalanche, το αρχείο αυτό βρίσκεται αρχικά σε έναν κεντρικό εξυπηρετητή. Καθώς όμως ο κεντρικός εξυπηρετητής έχει περιορισμένη ικανότητα στο να εξυπηρετήσει αρκετούς αιτούντες, οι κόμβοι συνεργάζονται μεταξύ τους για να διευκολύνουν τη διαδικασία. Όπως στο BitTorrent έτσι και εδώ, το αρχικό αρχείο χωρίζεται σε μικρότερα κομμάτια, τα οποία διαμοιράζονται ανάμεσα στους ενδιαφερόμενους. Κάθε κόμβος μπορεί να ανταλλάσσει κομμάτια του αρχείου μόνο με τους κόμβους της γειτονιάς στην οποία ανήκει.

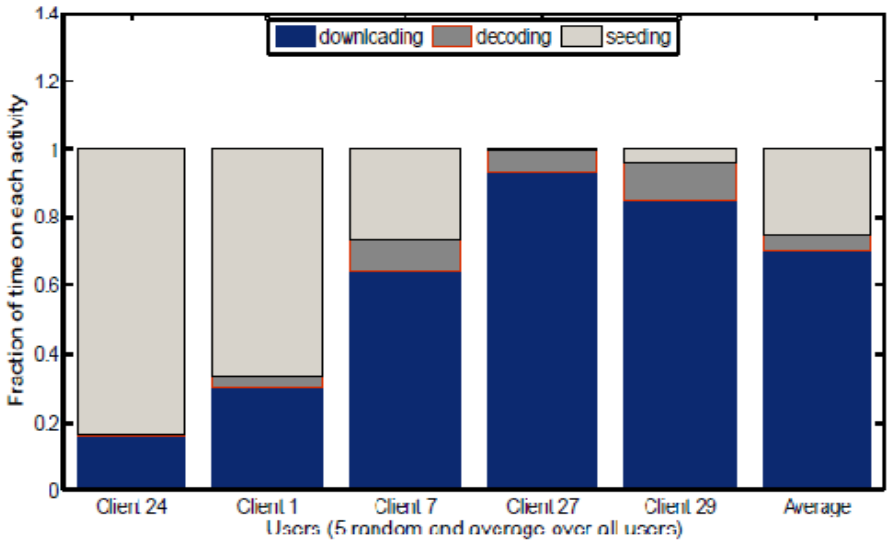
Πώς δημιουργείται αυτή η γειτονιά; Ένας κόμβος κατά την άφιξή του, επικοινωνεί με έναν κεντρικό εξυπηρετητή ο οποίος τον πληροφορεί για ένα τυχαίο υπαρκτό σύνολο κόμβων με τους οποίους μπορεί να συνδεθεί και να στήσει τη γειτονιά του. Αφού λοιπόν ολοκληρωθεί η διαδικασία αυτή μπορεί να αρχίσει η ανταλλαγή κομματιών. Σε περίπτωση που μειωθούν οι κόμβοι της γειτονιάς του, μπορεί να ζητήσει από τον κεντρικό εξυπηρετητή επιπλέον κόμβους για να συνδεθεί.

Όπως αναφέρθηκε το Avalanche χρησιμοποιεί την τεχνική κωδικοποίησης δικτύου. Έτσι, κάθε φορά που ένας κόμβος επιθυμεί να στείλει δεδομένα σε κάποιον άλλο κόμβο, παράγει έναν γραμμικό συνδυασμό όλων των κομματιών που κατέχει μέχρι εκείνη τη στιγμή και τα στέλνει. Η λειτουργία αυτή φαίνεται στο Σχήμα 3. Ο κόμβος Α ζητάει το αρχείο από τον εξυπηρετητή. Ο εξυπηρετητής ο οποίος κατέχει ολόκληρο το αρχείο, χωρίζει το αρχείο σε n κομμάτια: B_1 , B_2, ...,B_n και πολλαπλασιάζει κάθε κομμάτι B_i με κάποιους τυχαίους συντελεστές c_1 ,c_2 ,... ,c_n αντίστοιχα. Ύστερα, προσθέτει όλους τους πολλαπλασιασμούς και σχηματίζει το τελικό μπλοκ E_1, όπου $E_1 = c_1 B_1 + c_2 B_2 + ... + c_n B_n$. Έστω ότι ο Α έχει ακόμα ένα κωδικοποιημένο κομμάτι στην αποθήκη του το E_2, το οποίο έχει σχηματιστεί με τον ίδιο τρόπο όπως και το E_1, μόνο που τα κομμάτια B_1 , B_2, ...,B_n έχουν πολλαπλασιαστεί με ένα διαφορετικό διάνυσμα συντελεστών c'_1 ,c'_2 ,... ,c'_n. Οπότε $E_2 = c'_1 B_1 + c'_2 B_2 + ... + c'_n B_n$. Όπως ο εξυπηρετητής συνδύασε με έναν συγκεκριμένο τρόπο όλα τα κομμάτια που είχε για να τα στείλει στον κόμβο Α, έτσι και ο Α, όπως και κάθε άλλος ενδιάμεσος κόμβος, όταν του ζητηθεί να στείλει δεδομένα από έναν άλλο κόμβο έστω τον Β, θα πρέπει να επιλέξει τόσους τυχαίους συντελεστές όσα και τα κομμάτια που έχει να συνδυάσει. Έτσι, καθώς ο Α έχει τα κωδικοποιημένα κομμάτια E_1, E_2 χρειάζεται δύο τυχαίους συντελεστές , έστω c''_1 , c''_2 , με τους οποίους θα πολλαπλασιάσει τα E_1, E_2 και κατόπιν θα προσθέσει, με αποτέλεσμα να δημιουργηθεί ένα νέο κομμάτι το $E_3 = c''_1 E_1 + c''_2 E_2$ με το διάνυσμα συντελεστών του να είναι το $c''_1 c_1 + c''_2 c'_1$, $c''_1 c_2 + c''_2 c'_2$, ... Οι τυχαίοι αυτοί συντελεστές επιλέγονται από ένα πεπερασμένο πεδίο αριθμών, αρκετά μεγάλο όμως ώστε η πιθανότητα να επιλεχθούν ίδια ή γραμμικά εξαρτημένα διανύσματα συντελεστών να είναι πολύ μικρή.



Σχήμα 3: Τρόπος κωδικοποίησης στο σύστημα Avalanche

Η αποκωδικοποίηση γίνεται με τον τρόπο που λύνεται ένα γραμμικό σύστημα, όπως για παράδειγμα με τη μέθοδο Gauss. Η αποκωδικοποίηση είναι εφικτή όταν ο κόμβος έχει λάβει n κωδικοποιημένα κομμάτια με γραμμικά ανεξάρτητα διανύσματα συντελεστών, καθώς όπως και στη μέθοδο Gauss είναι ανέφικτο να επιλύσεις ένα γραμμικό σύστημα με γραμμικά εξαρτημένα διανύσματα συντελεστών. Έτσι, το πεπερασμένο πεδίο που αναφέρεται παραπάνω χρειάζεται να είναι αρκετά μεγάλο της τάξης του 2^{16} για να αποφευχθεί η γραμμική εξάρτηση. Η διαδικασία της αποκωδικοποίησης καταλαμβάνει το δικό της χρόνο στην όλη διαδικασία, το οποίο φαίνεται στο διάγραμμα στο Σχήμα 4, από ένα αντιπροσωπευτικό δείγμα κόμβων. Ο χρόνος της αποκωδικοποίησης δεν ξεπερνάει όμως το 6% κατά μέσο όρο της συνολικής διαδικασίας, σε πειράματα που έγιναν σε πραγματικό περιβάλλον.



Σχήμα 4: Ποσοστό χρόνου που καταλαμβάνει η κάθε λειτουργία.

Ενώ στο διαμοιρασμό του αρχείου χωρίς κωδικοποίηση δικτύου εφαρμόζονται πολιτικές (όπως πιο σπάνιο κομμάτι) όσο αφορά την επιλογή του κομματιού για αποστολή-λήψη, στην περίπτωση του Avalanche με κωδικοποίηση δικτύου ανάμεσα σε όλους του κόμβους, δε χρειάζεται κάποια τέτοια πολιτική. Κάθε κόμβος συνδυάζει σε ένα όλα τα κομμάτια που έχει και το στέλνει στον ενδιαφερόμενο. Αν κάποιος κόμβος λάβει από έναν άλλον ένα κομμάτι το οποίο έχει όμοιους συντελεστές με κάποιο από αυτά που κατέχει ήδη ο παραλήπτης τότε ο παραλήπτης υποθέτει ότι ο συγκεκριμένος γείτονάς του δεν έχει κάτι νέο να του προσφέρει και δεν τον «ξαναενοχλεί» μέχρις ότου ο γείτονας λάβει κάποιο καινούργιο.

Όσο αφορά τα κίνητρα που δίνονται στους χρήστες, να συνεισφέρουν στο διαμοιρασμό του αρχείου ανεβάζοντας δεδομένα, χρησιμοποιείται μια τεχνική παρόμοια με την tit-for-tat του BitTorrent, όπου ο χρήστης δε στέλνει δεδομένα σε κάποιον άλλον, αν και ο άλλος δε συνεργάζεται στέλνοντας δεδομένα.

Επίσης, κάποια ακόμα γεγονότα τα οποία οδηγούν στην ισχυριζόμενα καλύτερη επίδοση του Avalanche είναι ότι:

- Οι κόμβοι του δικτύου θα ολοκληρώσουν τη λήψη του αρχείου σε περίπτωση που ο αρχικός εξυπηρετητής και μοναδικός seeder στο δίκτυο αποχωρήσει. Αρκεί να έχει ανεβάσει τουλάχιστον τόσα γραμμικά ανεξάρτητα κομμάτια όσα και τα κομμάτια του αρχείου.

- Οι κόμβοι δε χρειάζεται να γνωρίζουν τι έχουν λάβει οι υπόλοιποι κόμβοι για να αποφασίσουν ποιο είναι το καλύτερο να αποκτήσουν, κάτι το οποίο κάνει πιο εύκολη τη λειτουργία όσο αφορά το συγκεκριμένο τμήμα λειτουργικότητας.

Πέρα όμως των θετικών στοιχείων που μπορεί να έχει το νέο αυτό σύστημα υπάρχουν και κάποια πράγματα τα οποία χρήζουν προσοχή, όπως είναι το θέμα της

ασφάλειας. Το θέμα αυτό προκύπτει, από τη στιγμή που κάθε ενδιάμεσος κόμβος, διαμορφώνει το μπλοκ μέσω της κωδικοποίησης, που θα στείλει σε κάποιον άλλον. Έτσι, ένας κακόβουλος χρήστης θα μπορούσε να διαμορφώσει ένα μπλοκ με λάθος δεδομένα και να προκαλεί προβλήματα στην ολοκλήρωση της διαδικασίας.

Στα συστήματα που ανταλλάσσουν δεδομένα χωρίς κωδικοποίηση όπως το BitTorrent πρωτόκολλο, η πληροφορία που λαμβάνεται μπορεί να επαληθευτεί αν είναι αυθεντική ή εάν έχει υποστεί μεταβολή με τη βοήθεια συναρτήσεων κατακερματισμού. Κάθε κομμάτι του αρχείου έχει το δικό του κλειδί αυθεντικότητας το οποίο μπορεί να επαληθευτεί κατά την ολοκλήρωση λήψης του κομματιού, καθώς επίσης και στο τέλος για όλο το αρχείο, όταν αυτό θα έχει ληφθεί πλήρως.

Αλγόριθμος XOR Κωδικοποίησης Δικτύου Σε Δίκτυο Ομότιμων Κόμβων

Στο κεφάλαιο αυτό θα παρουσιάσουμε τον προτεινόμενο αλγόριθμο διαμοιρασμού ενός αρχείου σε ένα δίκτυο ομότιμων κόμβων ο οποίος βασίζεται στην χρήση της τεχνικής *γραμμικής xor - κωδικοποίησης μπλοκ* αρχείου και συνδυάζεται με την συνεχή παρατήρηση των περιεχομένων της γειτονιάς ενός κόμβου. Ο αλγόριθμος της ανταλλαγής κωδικοποιημένων δεδομένων μεταξύ των κόμβων είναι υβριδικός με την έννοια ότι δε βασίζεται εξ ολοκλήρου στην κωδικοποίηση δικτύου για όλα τα μπλοκ αλλά συνυπάρχει με τον μη κωδικοποιημένο τρόπο διαμοιρασμού.

Σύστημα χωρίς κωδικοποίηση

Αρχικά, το δίκτυο αποτελείται από κόμβους οι οποίοι διαθέτουν ολόκληρο το αρχείο και αποκαλούνται seeders, καθώς και από κόμβους οι οποίοι δεν έχουν ολόκληρο το αρχείο αποκαλούμενοι leechers. Όπως περιγράφεται και παραπάνω στο BitTorrent πρωτόκολλο, οι leechers είναι αυτοί που επιθυμούν να αποκτήσουν το αρχείο ζητώντας δεδομένα από τους υπόλοιπους κόμβους της γειτονιάς τους, ενώ παράλληλα όσο υπάρχουν στο δίκτυο συμμετέχουν στο διαμοιρασμό του αρχείου στέλνοντας δεδομένα, με βάση αυτά που ήδη έχουν.

Οι τύποι μηνυμάτων που ανταλλάσσουν μεταξύ τους οι κόμβοι χωρίς κάποια κωδικοποίηση είναι οι :

- **REQ**(request): μήνυμα αίτησης για την παραλαβή ενός μπλοκ αρχείου. Επισυνάπτεται πληροφορία των id που κατέχει ο κόμβος που στέλνει το μήνυμα.
- **AVL**(available): ο κόμβος έχει διαθέσιμο ένα μπλοκ αρχείου για αποστολή στον κόμβο από τον οποίο έλαβε ένα μήνυμα τύπου REQ.
- **NOT_AVL**(not available): ο κόμβος δεν έχει διαθέσιμο κάποιο χρήσιμο μπλοκ αρχείου για τον κόμβο από τον οποίο έλαβε ένα μήνυμα τύπου REQ.

Οι κόμβοι είναι επιφορτισμένοι με τις ακόλουθες λειτουργίες:

- **Leecher**
  - Στέλνει περιοδικά αιτήσεις-REQ σε τυχαίους κόμβους της γειτονιάς, επισυνάπτοντας στο μήνυμα την πληροφορία για το ποια μπλοκ αρχείου κατέχει μέχρι εκείνη τη στιγμή.

- Δέχεται αιτήσεις-REQ, επεξεργάζεται την επισυναπτόμενη πληροφορία του μηνύματος αίτησης και απαντά εάν έχει-AVL, με ένα τυχαίο μπλοκ αρχείου διαφορετικό από αυτά που επισυνάπτει ο αιτών. Εάν δεν έχει κάποιο χρήσιμο μπλοκ για τον αιτών του απαντά με NOT_AVL.

➢ **Seeder**

- Δέχεται αιτήσεις-REQ, επεξεργάζεται την επισυναπτόμενη πληροφορία του μηνύματος αίτησης και απαντά πάντα (αφού έχει όλο το αρχείο) με ένα τυχαίο μπλοκ διαφορετικό από αυτά που επισυνάπτει ο αιτών στο μήνυμα της αίτησης.

Αρχικά, όταν ξεκινάει η διαδικασία ο leecher στέλνει μία αίτηση πάντα σε έναν seeder για να αποκτήσει όσο το δυνατόν γρηγορότερα κάποιο μπλοκ αρχείου. Έτσι, ο leecher θα είναι σε θέση να παίζει και το ρόλο του διαμοιραστή μετά την λήψη του πρώτου του μπλοκ. Να σημειωθεί ότι κάθε διαφορετικό μπλοκ του αρχείου έχει το δικό του id. Ύστερα από την πρώτη συνδιαλλαγή με τον κόμβο που έχει ολόκληρο το αρχείο, ο leecher στέλνει αιτήσεις σε τυχαίους κόμβους της γειτονιάς του. Έτσι, με την επικοινωνία μεταξύ των κόμβων του δικτύου, το αρχείο διασπείρεται σε όλο το δίκτυο και στο τέλος όλοι οι κόμβοι θα έχουν από ένα αντίγραφό του.

Σύστημα με XOR - κωδικοποίηση

Το σύστημα με την xor κωδικοποίηση δικτύου που υλοποιήθηκε είναι στην ουσία επέκταση του συστήματος που περιγράφηκε παραπάνω, αυτό του χωρίς κωδικοποίηση. Βασίζεται λοιπόν πάνω σε αυτό, έχοντας ενσωματώσει κάποιες επιπλέον λειτουργίες. Έτσι, ο διαμοιρασμός του αρχείου γίνεται ανταλλάσσοντας δεδομένα με τη βοήθεια των βασικών μηνυμάτων του πρωτοκόλλου REQ, AVL και NOT_AVL, ενώ παράλληλα ανά κάποια σταθερή χρονική περίοδο της τάξης των 20 δευτερολέπτων, οι κόμβοι οι οποίοι δεν έχουν ολοκληρώσει το αρχείο –οι λεγόμενοι leechers- στέλνουν ένα μήνυμα σε κάθε κόμβο της γειτονιάς τους, πληροφορώντας για το ποια αποκωδικοποιημένα μπλοκ κατέχουν μέχρι εκείνη τη στιγμή. Κάθε κόμβος παραλαμβάνει τα μηνύματα αυτά, κάνει τις κατάλληλες ενέργειες και στο τέλος αφού θα έχει παραλάβει το μήνυμα αυτό από κάθε leecher της γειτονιάς του, αποφασίζει να στείλει ένα xor-κωδικοποιημένο μπλοκ σε κάποιους κόμβους από τους οποίους έλαβε αυτό το μήνυμα.

Οι επιπλέον τύποι μηνυμάτων που στέλνονται από τους κόμβους είναι :

- **INFO** (information): μήνυμα το οποίο περιέχει την πληροφορία για το ποια αποκωδικοποιημένα μπλοκ κατέχει μέχρι εκείνη τη στιγμή, ο κόμβος που το στέλνει. Στην ορολογία τον συστημάτων ομότιμων κόμβων χρησιμοποιείται επίσης ο όρος buffer maps.

- **XOR**: το μήνυμα αυτό περιέχει κωδικοποιημένα δεδομένα με χρήση της xor λειτουργίας και στέλνεται από τον κόμβο σε ένας μέρος των γειτόνων του.

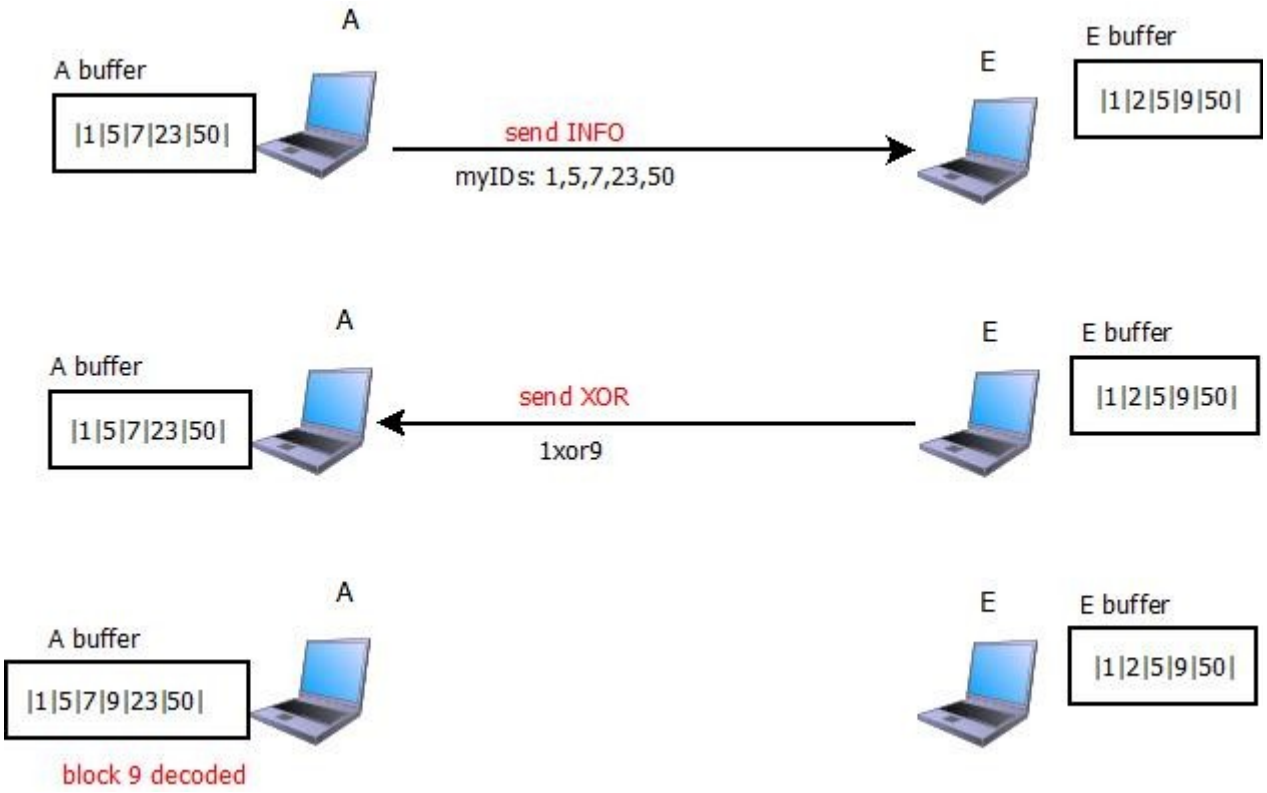
Επίσης, οι επιπλέον λειτουργίες με τις οποίες είναι επιφορτισμένοι οι κόμβοι του δικτύου είναι οι εξής:

- **Leecher**
  - Στέλνει INFO μήνυμα σε όλους τους κόμβους της γειτονιάς του κάθε μία συγκεκριμένη χρονική περίοδο.

  - Λαμβάνει INFO μηνύματα, συγκεντρώνει και αξιοποιεί την πληροφορία με αποτέλεσμα να κωδικοποιεί μπλοκ και να στέλνει XOR μηνύματα στους γείτονες του.

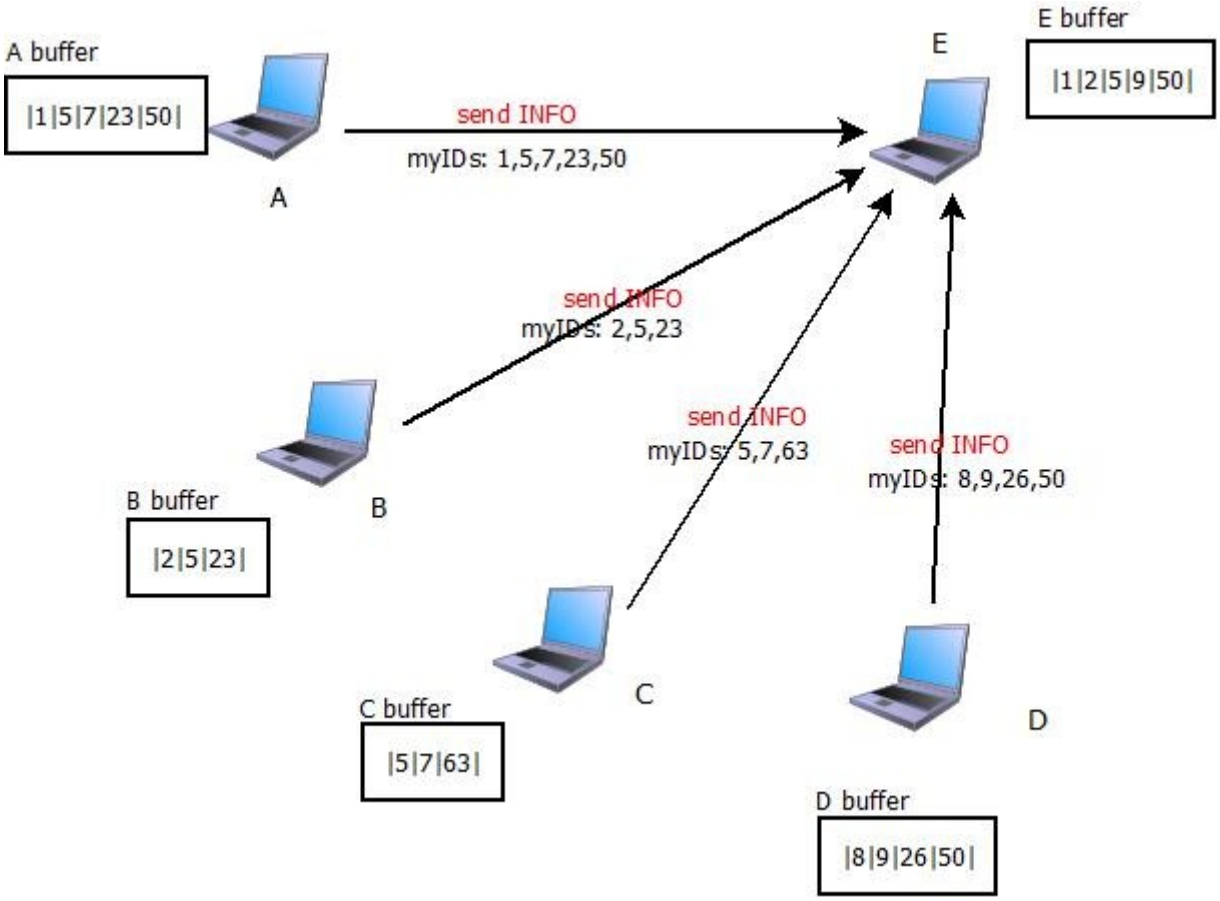
- **Seeder**
  - Λαμβάνει INFO μηνύματα, συγκεντρώνει και αξιοποιεί την πληροφορία με αποτέλεσμα να κωδικοποιεί μπλοκ και να στέλνει XOR μηνύματα στους γείτονες του.

Για να αποφασίσει ο κόμβος ποια μπλοκ θα κωδικοποιήσει και σε ποιους κόμβους θα στείλει το κωδικοποιημένο μπλοκ, πρέπει να χειριστεί κατάλληλα την πληροφορία που αποκτά μέσω των INFO μηνυμάτων. Κάποια μπλοκ του κόμβου που στέλνει INFO, με τα μπλοκ του κόμβου που λαμβάνει το μήνυμα, υπάρχει πιθανότητα να είναι κοινά και στους δύο. Ενώ, κάποια άλλα μπλοκ που έχει ο κόμβος που έλαβε INFO, υπάρχει πιθανότητα να μην υπάρχουν σε αυτόν που έστειλε. Οπότε, μπορεί να συνδυαστεί ένα μπλοκ το οποίο έχει ήδη, μαζί με ένα μπλοκ που του λείπει και μόλις λάβει το κωδικοποιημένο μπλοκ να είναι σε θέση να το αποκωδικοποιήσει και να ανακτήσει αυτό που δεν είχε, όπως φαίνεται στο Σχήμα 5.



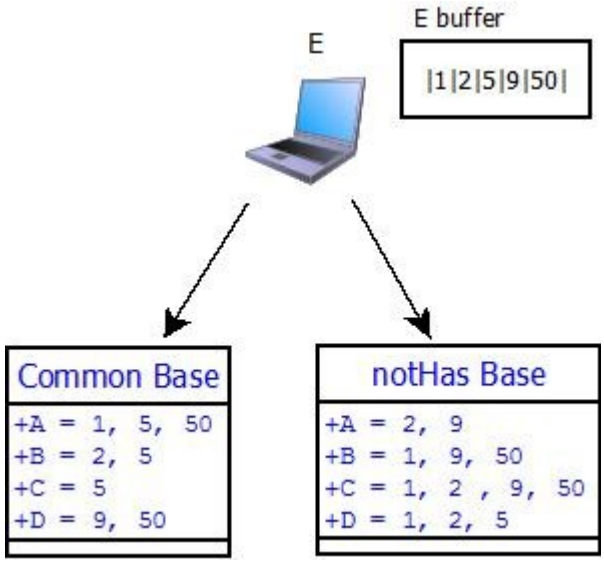
Σχήμα 5: Απλό παράδειγμα ανταλλαγής κωδικοποιημένων δεδομένων ανάμεσα σε δύο κόμβους.

Στη γενική περίπτωση τώρα, όπου έχουμε παραπάνω από δύο κόμβους, κάθε κόμβος ενημερώνει δύο δομές δεδομένων κάθε φορά που λαμβάνει ένα μήνυμα τύπου INFO από κάποιον γείτονά του. Στη μία δομή, την οποία ονομάζουμε Common, αποθηκεύει τα κοινά id μεταξύ αυτού και του κόμβου που στέλνει το INFO μήνυμα. Στην άλλη δομή η οποία ονομάζεται notHas, αποθηκεύονται τα id που έχει ο κόμβος που λαμβάνει INFO και δεν έχει ο γειτονικός κόμβος που του έστειλε μήνυμα τύπου INFO. Ας υποθέσουμε ότι έχουμε τη γειτονιά που φαίνεται στο Σχήμα 6 και ο κόμβος E λαμβάνει μηνύματα τύπου INFO από τους A, B, C, D. Κάθε κόμβος από τους A, B, C, D επισυνάπτει στο μήνυμα INFO τα id των μπλοκ τα οποία έχει μέχρι εκείνη τη στιγμή.



Σχήμα 6: Αποστολή μηνύματος πληροφορίας από leechers.

Ο κόμβος Ε λαμβάνει αυτά τα μηνύματα τα οποία επεξεργάζεται ένα προς ένα και φτιάχνει τις δομές Common και notHas οι οποίες καταλήγουν στην μορφή που φαίνεται στο Σχήμα 7. Στην συνέχεια συγκρίνει τα id κάθε κόμβου με τα δικά του. Όποια id είναι κοινά μεταξύ τους τα αποθηκεύει στη δομή Common μαζί με το αναγνωριστικό του κάθε κόμβου, ενώ όποια id δεν έχει ο γειτονικός του κόμβος αλλά τα έχει ο Ε τα αποθηκεύει στη δομή notHas.



Σχήμα 7: Κατασκευή δομών δεδομένων από τον κόμβο Ε.

Αφού ένας κόμβος λάβει τα μηνύματα INFO από όλους τους γειτονικούς κόμβους και έχει αποθηκεύσει τις κατάλληλες πληροφορίες στις δομές Common και notHas, αρχίζει τη διαδικασία απόφασης του συνδυασμού μπλοκ που θα κωδικοποιήσει και την επιλογή των κόμβων στους οποίους θα στείλει το συνδυασμό. Οι δύο δομές βοηθούν στη διαδικασία αυτή, καθώς ξέρουμε ποια μπλοκ έχουν οι γείτονες και ποια δεν έχουν. Έτσι, μπορούμε να κωδικοποιήσουμε κάποιο μπλοκ το οποίο λείπει από κάποιους γείτονες, με κάποιο το οποίο ξέρουμε ότι έχουν, έτσι ώστε να είναι σε θέση να το αποκωδικοποιήσουν.

Επιλογή Μπλοκ προς Κωδικοποίηση

Η επιλογή των βέλτιστων κομματιών που θα κωδικοποιηθούν αποτελεί ένα πολύ σημαντικό πρόβλημα σε κάθε είδους σύστημα το οποίο υλοποιεί την τεχνική κωδικοποίησης δικτύου. Στο προτεινόμενο σύστημα στόχος είναι το νέο μπλοκ που θα συμπεριληφθεί στο συνδυασμό και που θα αποσταλεί σε γειτονικούς κόμβους, να μην υπάρχει σε όσο το δυνατόν περισσότερους από αυτούς. Κατά αυτόν τον τρόπο, επιλέγεται το id εκείνο το οποίο εμφανίζεται τις περισσότερες φορές στη δομή notHas και παράλληλα κρατάμε τους κόμβους από τους οποίους λείπει. Στην συνέχεια, για τους κόμβους αυτούς, αναζητούμε στη δομή Common το id το οποίο είναι περισσότερο κοινό μεταξύ τους. Για την επιλογή του πιο κοινού id μεταξύ των συγκεκριμένων κόμβων, κάνουμε χρήση μιας μετρικής η οποία παίρνει τιμές από το 0 ως το 1. Στην ουσία, με αυτή τη μετρική ελέγχεται το ποσοστό των κόμβων για τους οποίους είμαστε σίγουροι ότι θα μπορούν να αποκωδικοποιήσουν το κωδικοποιημένο μπλοκ, λόγω του ότι ήδη έχουν ένα άλλο μπλοκ. Δηλαδή, αν η τιμή της μετρικής είναι 0.5 και σταλεί ο συνδυασμός στους κόμβους, για τους οποίους βρέθηκε η μέγιστη εμφάνιση id στη βάση notHas, τότε ξέρου-

με ότι τουλάχιστον οι μισοί κόμβοι –αποδέκτες του κωδικοποιημένου μηνύματος- θα μπορούν να αποκωδικοποιήσουν και να ανακτήσουν το νέο μπλοκ. Γενικότερα, αν N είναι το πλήθος των κόμβων που έχουμε βρει ότι τους λείπει ένα συγκεκριμένο μπλοκ και M κόμβοι από τους N (με M<=N) έχουν μεταξύ τους κάποιο κοινό μπλοκ, τότε για να προχωρήσουμε στην αποστολή του κωδικοποιημένου συνδυασμού θέλουμε M >= μετρικής * N. Για να γίνει πιο κατανοητός ο παραπάνω αλγόριθμος παρουσιάζουμε ένα παράδειγμα στην συνέχεια.

Παράδειγμα

Πιο συγκεκριμένα, στο στιγμιότυπο των Σχημάτων 6 και 7 ο κόμβος E θα δει ότι:

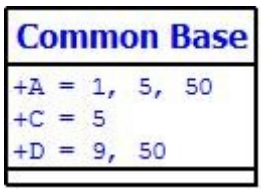
- Το id 2 λείπει από τους A, C, D
- Το id 9 λείπει από τους A, B, C
- Το id 1 λείπει από τους B, C, D
- Το id 50 λείπει από τους B, C
- Το id 5 λείπει από τους D

Τα id 2,9,1 είναι αυτά που λείπουν από ισάριθμο μέγιστο πλήθος κόμβων. Οπότε επιλέγουμε ένα από αυτά τυχαία.

Έστω ότι επιλέγουμε το id 2.

Τότε, για τους κόμβους A, C, D που αντιστοιχούν στο id 2, πάμε στη δομή Common να βρούμε το id το οποίο είναι περισσότερο κοινό μεταξύ τους.

Η δομή Common του κόμβου E για τους A, C, D έχει ως εξής:



- Το id 1 είναι κοινό στους A
- Το id 5 είναι κοινό στους A, C
- Το id 50 είναι κοινό στους A, D
- Το id 9 είναι κοινό στους D

Τα id 5, 50 είναι αυτά που είναι περισσότερο κοινά μεταξύ των κόμβων τα οποία εμφανίζονται σε δύο από τους τρεις κόμβους, επιλέγουμε ένα έστω το id 5.

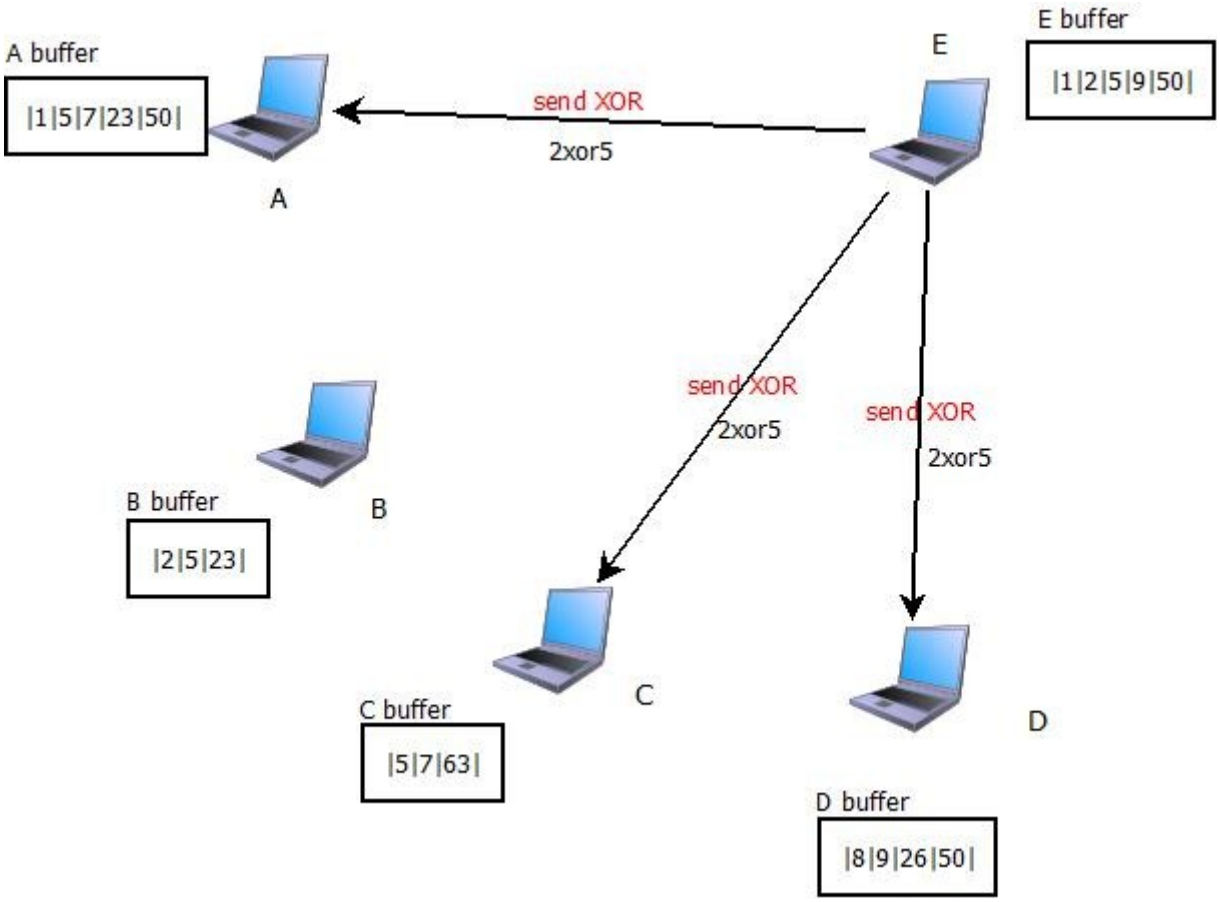
Έστω ότι η τιμή της μετρικής είναι 0.5.

Το πλήθος των κόμβων που επιθυμεί ο E να στείλει το συνδυασμό είναι 3: A, C, D.

Επίσης έχουμε ότι τιμή μετρικής * πλήθος κόμβων = 0.5 * 3 = 1.5 .

Συνεπώς το πλήθος κόμβων που έχουν το id 5 είναι 2 > 0.5 * 3.

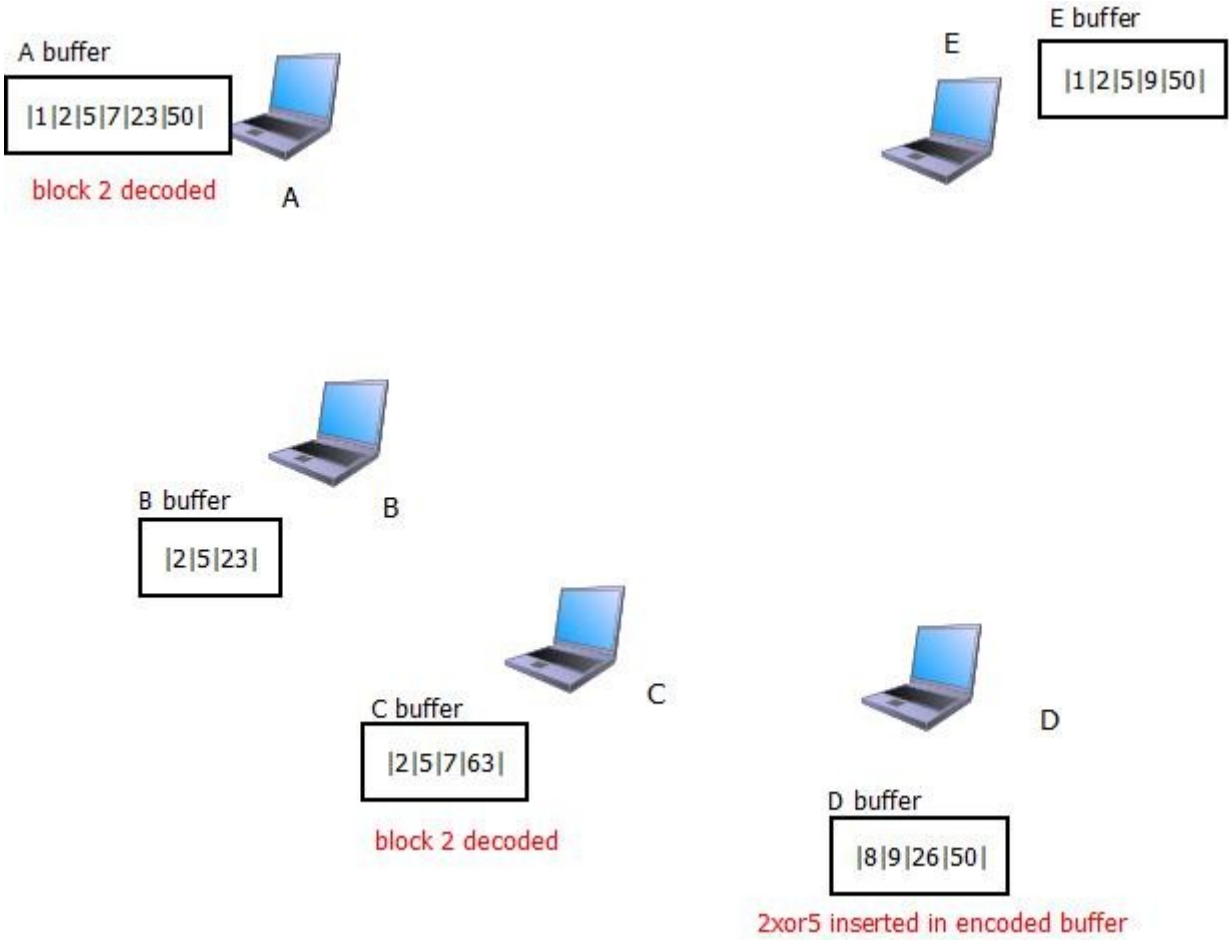
Αυτό σημαίνει ότι ικανοποιείται η συνθήκη της μετρικής, άρα στο συνδυασμό συμπεριλαμβάνουμε για να κωδικοποιήσουμε τα μπλοκ με id 5 & 2, το οποίο ακολούθως ο κόμβος E το στέλνει στους κόμβους A, C, D όπως φαίνεται και στο Σχήμα 8.



Σχήμα 8: Αποστολή κωδικοποιημένου μπλοκ.

Σε περίπτωση που αποτύχει η ικανοποίηση της συνθήκης με τη μετρική την πρώτη φορά, εκτελούμε την ίδια διαδικασία για το επόμενο περισσότερο εκλιπών id της δομής notHas του κόμβου.

Το κωδικοποιημένο μπλοκ παραλαμβάνεται από τους κόμβους A, C, D. Οι κόμβοι οι οποίοι έχουν το ένα από τα δύο του συνδυασμού μπορούν να αποκωδικοποιήσουν το μπλοκ, διαφορετικά εισάγουν το συνδυασμό στην αποθήκη με τα κωδικοποιημένα μπλοκ, Σχήμα 9.



Σχήμα 9: Παραλαβή κωδικοποιημένου μπλοκ και αποκωδικοποίηση του αν είναι εφικτό.

Η λειτουργία της xor είναι η εξής:

Είσοδος		**Έξοδος**
A	**B**	**AxorB**
0	0	0
0	1	1
1	0	1
1	1	0

Έστω ότι έχουμε δύο μπλοκ το X και το Y.

X: 01001

Y: 01011

Ας υποθέσουμε ότι κωδικοποιούμε τα δύο αυτά μπλοκ και αυτό που προκύπτει είναι το Z: X xor Y: 00010. Τότε ο κόμβος που θα παραλάβει το Z, εάν έχει ή το X ή το Y θα μπορεί να αποκωδικοποιήσει το Y ή το X αντίστοιχα, με την αντίστροφη διαδικασία που

ακολουθήθηκε για την κωδικοποίηση.

Κάθε φορά που κάποιος κόμβος λαμβάνει ένα νέο μπλοκ ή αποκωδικοποιεί κάποιο από την αποθήκη με τα κωδικοποιημένα μπλοκ, ελέγχει πάντα αν μπορεί να αποκωδικοποιήσει κάποιον άλλο συνδυασμό.

Επίσης, γίνεται η χρήση μετρικής, καθώς στέλνονται μπλοκ σε κόμβους οι οποίοι δεν έχουν απαραίτητα το μπλοκ εκείνο που θα τους βοηθήσει να αποκωδικοποιήσουν το συνδυασμό. Το πλεονέκτημα που έχουμε στέλνοντας σε κόμβους που δεν έχουν κανένα μπλοκ του συνδυασμού, είναι ότι αποκτούν 2 μπλοκ, τα οποία σίγουρα δεν είχαν μέχρι τη στιγμή που έστειλαν το μήνυμα τύπου INFO και είναι πιθανό να τα αποκωδικοποιήσουν στο άμεσο μέλλον, είτε λαμβάνοντας κάποιο επόμενο συνδυασμό που θα είναι σε θέση να αποκωδικοποιήσουν και έτσι να προκύψει κάποιο χρήσιμο για αλυσιδωτή αποκωδικοποίηση, είτε στο μεταξύ να λάβει κάποιο μπλοκ από αυτά του συνδυασμού σε απάντηση απλού request, είτε σε επόμενα requests τους. Οι παραπάνω υποθέσεις μπορούν να συνδυαστούν με την πιθανότητα ο κόμβος που θα επικοινωνήσουν, να είναι σε κοντινή απόσταση.

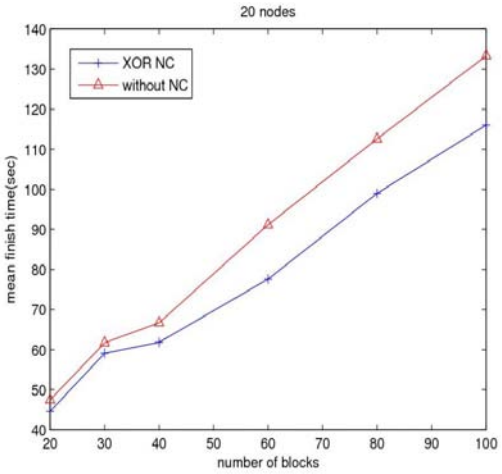
Πειραματικά Αποτελέσματα

Στην ενότητα αυτή παρουσιάζουμε πειραματικά αποτελέσματα των συστημάτων με κωδικοποίηση και χωρίς, τα οποία σκοπό έχουν να αναδείξουν τις μεταξύ τους διαφορές στη συμπεριφορά και στην επίδοση. Εξετάζεται η επίδραση στη συμπεριφορά για διαφορετικό μέγιστο δυνατό ρυθμό αποστολής και λήψης δεδομένων, για διαφορετικούς πληθυσμούς κόμβων και διαφορετικό πλήθος μπλοκ αρχείου. Τα πειράματα έγιναν με τη χρήση του προσομοιωτή Oversim [5], ο οποίος ειδικεύεται στην προσομοίωση δικτύων ομότιμων κόμβων.

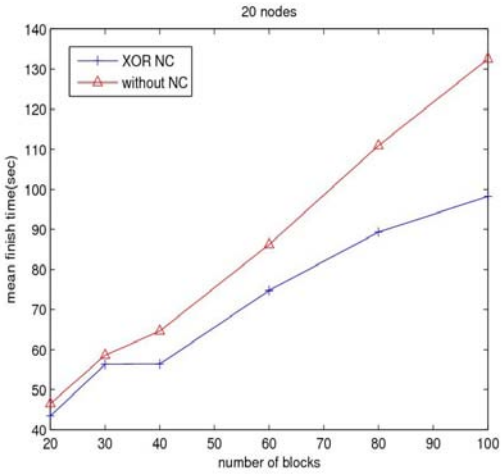
Πιο συγκεκριμένα, τα δύο συστήματα προσομοιώνονται στον ίδιο τύπο δικτύου, με τις ίδιες δηλαδή παραμέτρους και χαρακτηριστικά. Το δίκτυο της προσομοίωσης αποτελεί ένα ιδανικό δίκτυο στο οποίο δε χάνονται πακέτα. Χρησιμοποιείται το πρωτόκολλο μεταφοράς UDP το οποίο ναι μεν δεν επιλέγεται για μεταφορά αρχείων, αλλά στα πλαίσια της συγκριτικής προσομοίωσης και από τη στιγμή που δεν χάνονται πακέτα στο δίκτυο δε δημιουργείται πρόβλημα. Επίσης, θεωρούμε συμμετρικές συνδέσεις μεταξύ των κόμβων, με ίσο μέγεθος δυνατού ρυθμού αποστολής και λήψης δεδομένων. Κάθε κόμβος λαμβάνει εικονικά τη δική του θέση στο χώρο έχοντας τις δικές του συντεταγμένες σε χώρο δύο διαστάσεων. Η ευκλείδεια απόσταση μεταξύ των κόμβων επιδρά στην καθυστέρηση μεταφοράς δεδομένων μεταξύ τους. Το αρχείο που πρέπει να διαμοιραστεί, χωρίζεται σε μικρότερα μπλοκ, με κάθε ένα να έχει το δικό του αναγνωριστικό. Το μέγεθος του κάθε μπλοκ έχει οριστεί σε 16KBytes. Τα μηνύματα του πρωτοκόλλου 100 bytes. Κάθε leecher στέλνει αιτήσεις REQ κάθε 1 δευτερόλεπτο, ενώ κάθε 20 δευτερόλεπτα στέλνεται από κάθε leecher μήνυμα ενημέρωσης INFO για το ποια μπλοκ κατέχει ήδη, προς όλους. Η τιμή της μετρικής που χρησιμοποιήθηκε στα πειράματα είναι 0,5.

Αποτελέσματα μέσου χρόνου ολοκλήρωσης

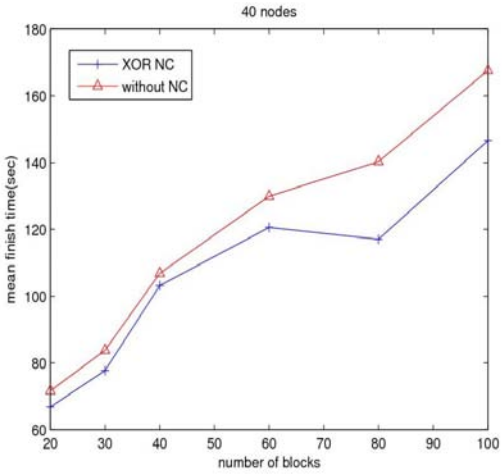
Αρχικά, παρουσιάζεται η διαφορά στον μέσο χρόνο ολοκλήρωσης του αρχείου από τους κόμβους,ανάμεσα στο σύστημα που χρησιμοποιεί κωδικοποίηση και αυτό χωρίς κωδικοποίηση. Ο μέσος χρόνος ολοκλήρωσης υπολογίζεται σύμφωνα με τους μεμονωμένους χρόνους ολοκλήρωσης του αρχείου από κάθε leecher της γειτονιάς. Τα αποτελέσματα που ακολουθούν έχουν προκύψει από προσομοιώσεις για διαφορετικό πλήθος κόμβων 20, 40, 60, 80, 100 κόμβους και διαφορετικό πλήθος μπλοκ για κάθε πλήθος κόμβων. Σε κάθε προσομοίωση στο δίκτυο υπάρχουν 3 seeders αρχικά και οι υπόλοιποι κόμβοι αποτελούν τους leechers (κάθε διαφορετική αναλογία θα αναφέρεται). Ο μέγιστος δυνατός ρυθμός αποστολής και λήψης δεδομένων είναι 1Mbps (αριστερή στήλη) και 10Mbps (δεξιά στήλη):



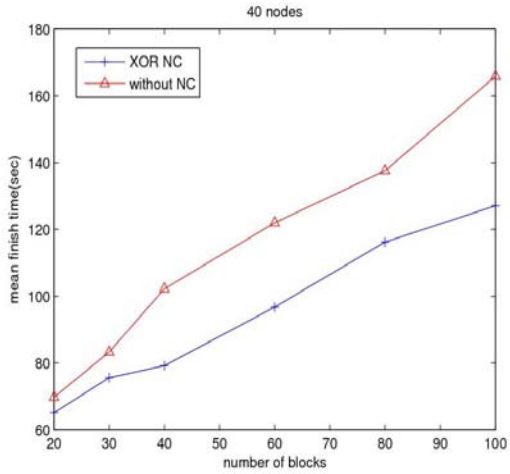
Σχήμα 10: 1 Mbps



Σχήμα 11: 10 Mbps

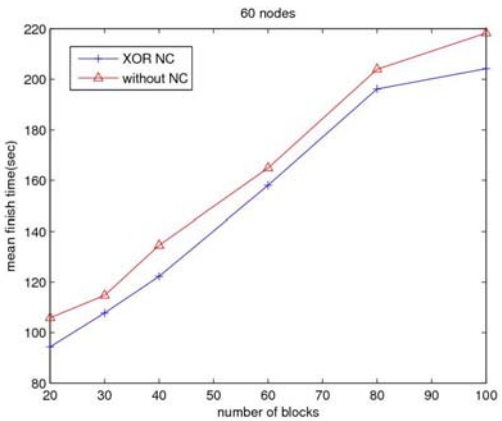


Σχήμα 12: 1 Mbps

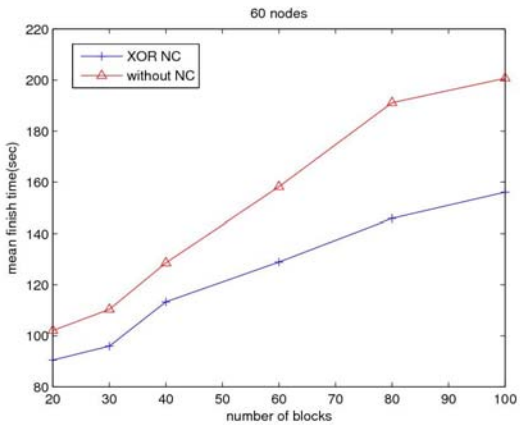


Σχήμα 13: 10 Mbps

Για μέγιστο δυνατό ρυθμό αποστολής 10 Mbps ο μέσος χρόνος ολοκλήρωσης γίνεται σταθερά μικρότερος σε σχέση με το 1Mbps, σε συνδυασμό με την αύξηση του αριθμού των μπλοκ του αρχείου.

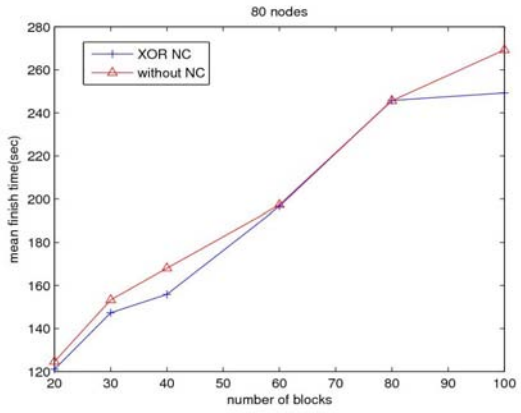


Σχήμα 14: 1 Mbps

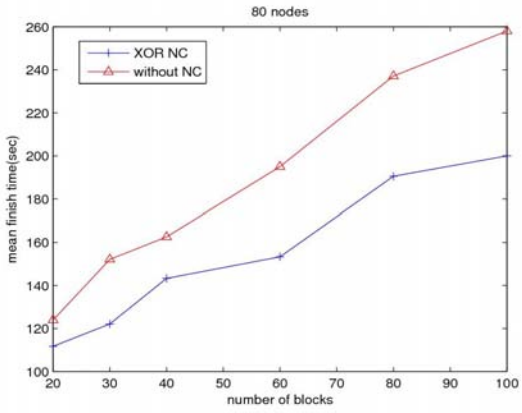


Σχήμα 15: 10 Mbps

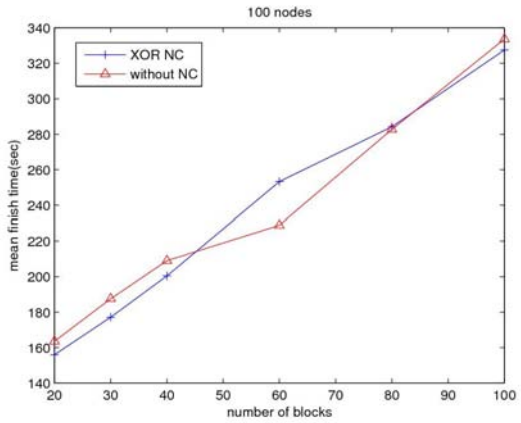
Όσο μεγαλώνει το πλήθος των κόμβων, για 1Mbps το σύστημα με την κωδικοποίηση δεν κλιμακώνει ικανοποιητικά σε σχέση με το σύστημα χωρίς κωδικοποίηση αλλά και σε σύγκριση με αυτό για 10 Mbps. Επίσης, δεν παρατηρούνται έντονες διαφορές στους μέσους χρόνους ολοκλήρωσης των κόμβων του συστήματος χωρίς κωδικοποίηση, καθώς δεν έχει «εκρήξεις» αποστολών σε πολλαπλούς κόμβους και το 1Mbps δεν αποτελεί εμπόδιο για την ομαλή διεξαγωγή της διαδικασίας.



Σχήμα 16: 1 Mbps



Σχήμα 17: 10 Mbps



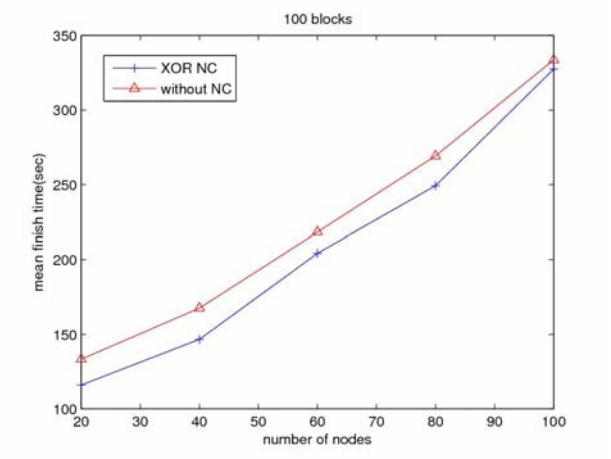
Σχήμα 18: 1 Mbps

Για 1 Mbps ο χρόνος μετάδοσης ενός μπλοκ απαιτεί 16KB / 1Mbps = 0,125 seconds. Αυτό σημαίνει ότι ένας κόμβος μπορεί να μεταδώσει 1/0,125 = 8 μπλοκ/sec. Αντίστοιχα για 10 Mbps ένας κόμβος μπορεί να μεταδώσει 80 μπλοκ/sec.

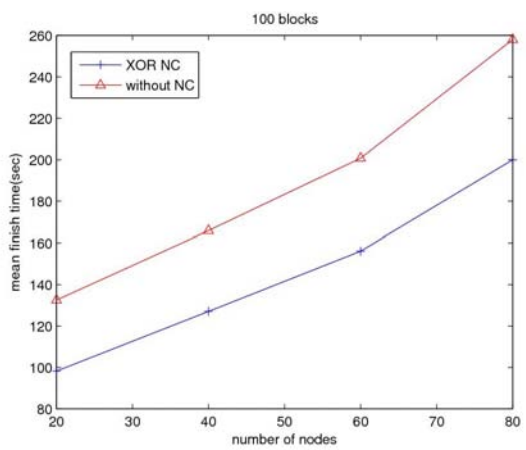
Όσο αυξάνεται λοιπόν ο ρυθμός αποστολής και λήψης δεδομένων, οι μεταδόσεις γίνονται γρηγορότερα. Ο κόμβος ο οποίος στέλνει το κωδικοποιημένο μήνυμα σε πολλαπλούς leechers, μπορεί να στείλει περισσότερα μπλοκ/sec και αντίστοιχα ο παραλήπτης να λάβει από πολλούς. Όσο αυξάνονται οι κόμβοι στη γειτονιά, αυξάνεται και το πλήθος των πολλαπλών leechers στους οποίους έναν κόμβος θα πρέπει να στείλει κωδικοποιημένο μήνυμα. Η διαδικασία αυτή ευνοείται όσο μεγαλύτερο είναι το πλήθος των μπλοκ που μπορεί να στείλει και να λάβει ένας κόμβος. Έτσι, επιτυγχάνονται καλύτεροι μέσοι χρόνοι ολοκλήρωσης καθώς αυξάνεται το πλήθος των κόμβων για

αυξημένο bandwidth.

Επίσης, οι διαφορές μεταξύ των δύο συστημάτων γίνονται περισσότερο εμφανείς καθώς ο χρόνος ολοκλήρωσης αυξάνει και η λειτουργία της κωδικοποίησης συμμετέχει εντονότερα. Στα παρακάτω διαγράμματα φαίνεται ο μέσος χρόνος ολοκλήρωσης για σταθερό πλήθος μπλοκ και διαφορετικό πλήθος κόμβων:

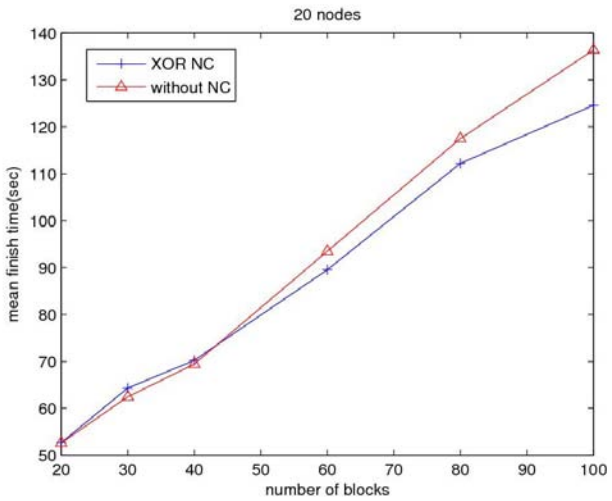


Σχήμα 19: 1 Mbps

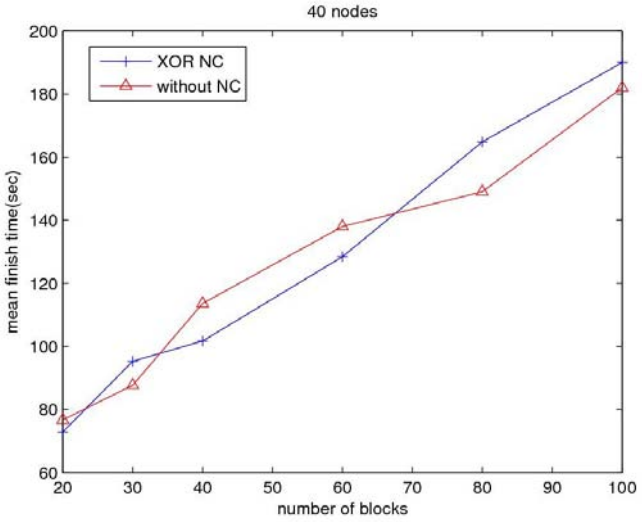


Σχήμα 20: 10 Mbps

Σύμφωνα με τα παραπάνω, όσο μειώνεται ο δυνατός ρυθμός αποστολής-λήψης και αυξάνεται το πλήθος των κόμβων η απόδοση του συστήματος με την κωδικοποίηση θα αποσταθεροποιείται, όπως φαίνεται και στα παρακάτω για 20 και 40 κόμβους με μέγιστο ρυθμό 500Kbps.



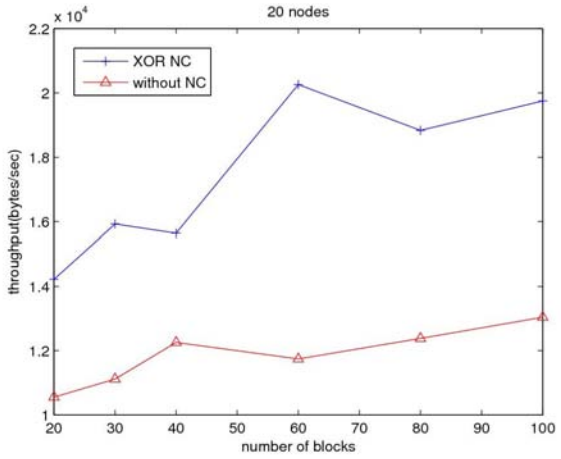
Σχήμα 21: 500Kbps



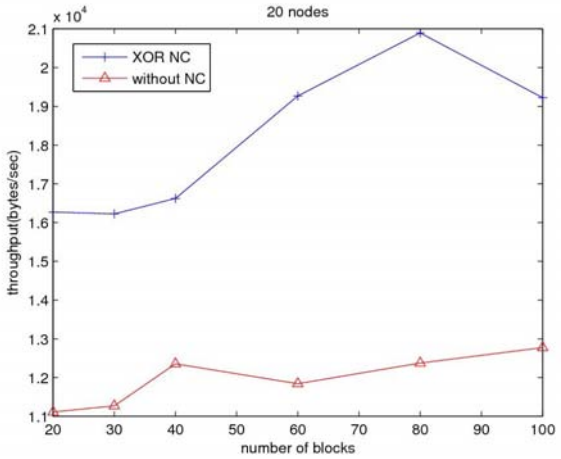
Σχήμα 22: 500Kbps

Αποτελέσματα Throughput

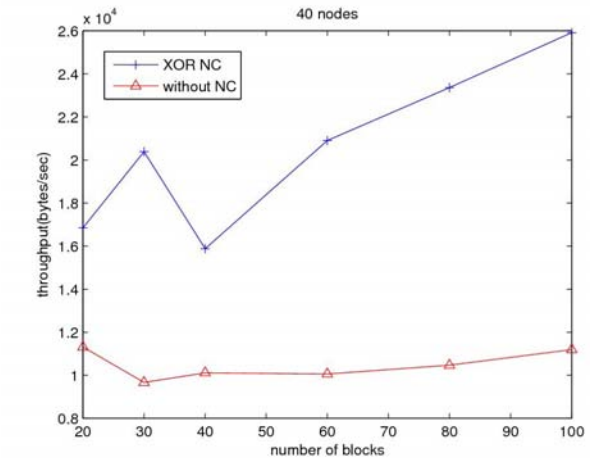
Στο σύστημα με την κωδικοποίηση στέλνονται περισσότερα δεδομένα με αποτέλεσμα την ευρύτερη χρήση των πόρων του δικτύου. Υπολογίζουμε το μέσο αριθμό Bytes/sec (throughput) που λαμβάνουν οι κόμβοι, για το μέσο χρόνο ολοκλήρωσης του αρχείου από όλους τους κόμβους, σύμφωνα με το μέσο πλήθος μπλοκ και άλλων μηνυμάτων του πρωτοκόλλου που λαμβάνονται.



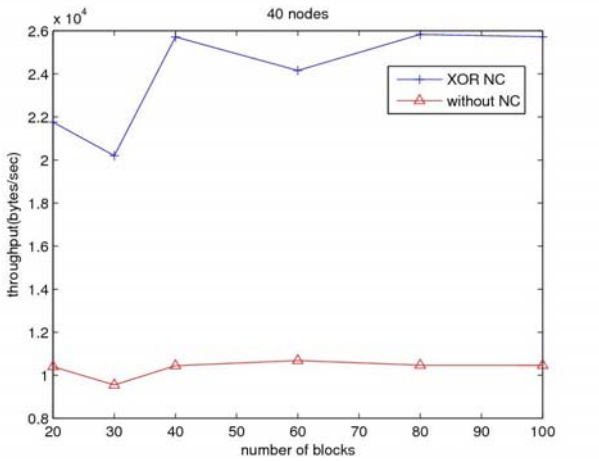
Σχήμα 23: 1 Mbps



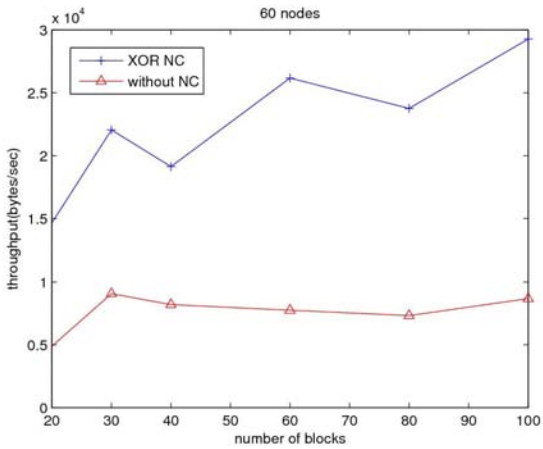
Σχήμα 24: 10 Mbps



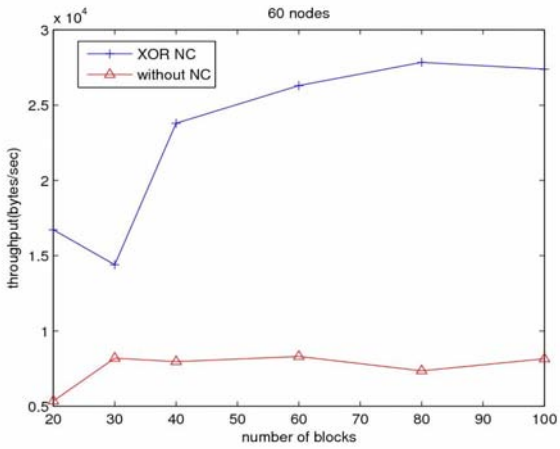
Σχήμα 25: 1 Mbps



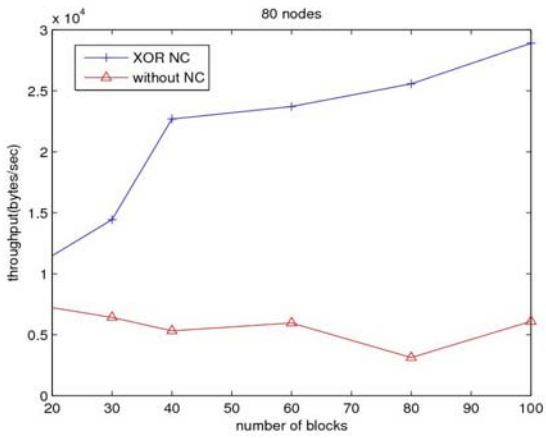
Σχήμα 26: 10 Mbps



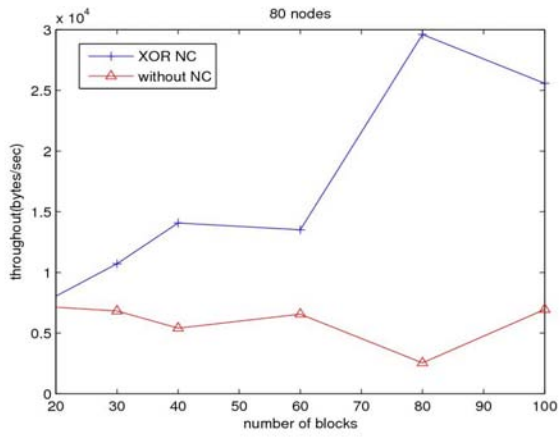
Σχήμα 27: 1 Mbps



Σχήμα 28: 10 Mbps



Σχήμα 29: 1 Mbps



Σχήμα 30: 10 Mbps

Όπως ήταν αναμενόμενο και φαίνεται στα διαγράμματα ο ρυθμός λήψης δεδομένων είναι μεγαλύτερος στο σύστημα με κωδικοποίηση.

Συγκριτικά για 1 και 10 Mbps το throughput δε διαφέρει στο σύστημα χωρίς κωδικοποίηση, καθώς όπως αναφέρθηκε και παραπάνω, δεν υπάρχουν «εκρήξεις» αποστολών σε πολλαπλούς χρήστες και οι αιτήσεις από τους κόμβους στέλνονται με σταθερό ρυθμό, με το 1Mbps να μην αποτελεί εμπόδιο σε σχέση με τα 10Mbps.

Όσο αφορά το σύστημα με κωδικοποίηση, συγκρίνοντας τις τιμές του throughput για ίδιους συνδυασμούς κόμβων – μπλοκ παρατηρούνται ορισμένες φορές χαμηλότερες

τιμές για μέγιστο ρυθμό αποστολής και λήψης 10Mbps σε σύγκριση με αυτές για 1Mbps. Όπως για παράδειγμα στο Σχήμα 29, 30 για πλήθος μπλοκ 60, η τιμή του throughput για 10Mbps είναι μικρότερη της αντίστοιχης για 1 Mbps. Για 10 Mbps, στο συγκεκριμένο συνδυασμό έχουμε μικρότερο μέσο χρόνο ολοκλήρωσης απ' ότι για 1 Mbps όπως φαίνεται στα Σχήματα 16, 17. Από τη στιγμή που το μέγεθος του αρχείου είναι σταθερό, άρα και το πλήθος των μπλοκ που πρέπει να λάβει ένας κόμβος για να το ολοκληρώσει είναι σταθερό, θα ήταν λογικό για 10 Mbps να έχουμε μεγαλύτερο throughput. Όμως, για το συνδυασμό 80-60, χρειάστηκε να σταλθούν περισσότερα μπλοκ μέχρι την ολοκλήρωση του αρχείου στην προσομοίωση για 1 Mbps συγκριτικά με αυτή των 10 Mbps και αυτό γιατί το πλήθος των κωδικοποιημένων άχρηστων μπλοκ που έλαβε ήταν μεγαλύτερο.

Ποσοστά χρήσιμων μπλοκ

Μια σημαντική παράμετρος του προτεινόμενου αλγορίθμου, αλλά και γενικά των αλγορίθμων κωδικοποίησης δικτύου, είναι ο καθορισμός της χρησιμότητας του κάθε κωδικοποιημένου πακέτου στην διαδικασία αποκωδικοποίησης. Στα συστήματα που μελετήσαμε, μπορούμε να διακρίνουμε τα μπλοκ σε «χρήσιμα» (innovative στην ορολογία της κωδικοποίησης δικτύου) και σε «άχρηστα» που λαμβάνει ο εκάστοτε κόμβος. Πιο συγκεκριμένα:

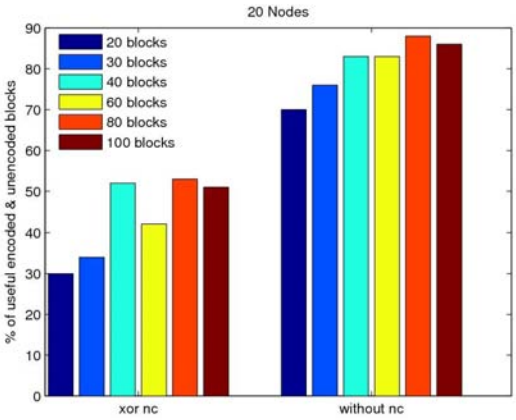
- Χρήσιμο μπλοκ χαρακτηρίζεται αυτό το οποίο λαμβάνεται για πρώτη φορά από τον κόμβο και μπορεί να αξιοποιηθεί από τον χρήστη ως προς την ολοκλήρωση του αρχείου.
- Άχρηστο χαρακτηρίζεται το μπλοκ το οποίο έχει ληφθεί ήδη στο παρελθόν και μια νέα λήψη του δεν προσφέρει κάτι ως προς την ολοκλήρωση του αρχείου, παρά μόνο καταναλώνει πόρους του δικτύου.

Υπάρχει πιθανότητα ένας χρήστης να λάβει το ίδιο μπλοκ από περισσότερους από έναν κόμβους, ως απάντηση σε μία αίτηση που πρόλαβε να στείλει περισσότερες από μία φορές προτού λάβει απάντηση με κάποιο διαθέσιμο μπλοκ. Επίσης, είναι πιθανό να στείλει κάποιες αιτήσεις, να λάβει κάποια διαθέσιμα μπλοκ ολοκληρώνοντας το αρχείο και να γίνει seeder. Οπότε κάθε επόμενο μπλοκ που θα λαμβάνει ως απάντηση στις αιτήσεις του το αγνοεί και έτσι χαρακτηρίζεται άχρηστο.

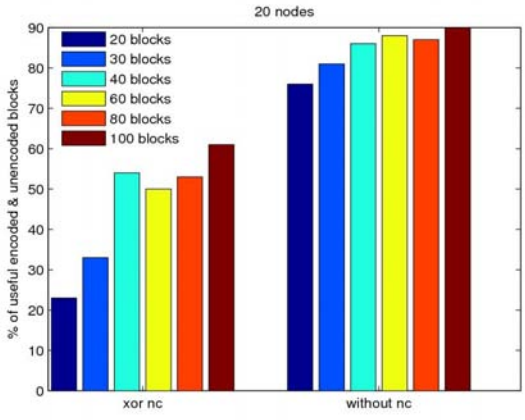
Η διαδικασία κωδικοποίησης περιέχει μεγαλύτερη πολυπλοκότητα, αφού χρειάζεται να ενημερωθούν όλοι κόμβοι του δικτύου από όλους τους leechers. Επίσης, χρειάζεται περισσότερο χρόνο για να λάβει μηνύματα από όλους τους κόμβους, να επιλέξει τα μπλοκ που θα κωδικοποιήσει και κατόπιν να τα στείλει σε πολλαπλούς κόμβους. Στο διάστημα που μεσολαβεί από τη στιγμή που ο κόμβος έστειλε INFO μήνυμα μέχρι να λάβει το κωδικοποιημένο μήνυμα, ο κόμβος λαμβάνει απαντήσεις σε προηγούμενες αιτήσεις του καλύπτοντας τα κενά του μη ολοκληρωμένου αρχείου. Συνεπώς, για κάθε κωδικοποιημένο μήνυμα που λαμβάνει, υπάρχει πιθανότητα να έχει και τα δύο μπλοκ του κωδικοποιημένου συνδυασμού και έτσι να είναι άχρηστη πληροφορία. Όσα περισσότερα μπλοκ μπορεί να στείλει ένας κόμβος προς έναν άλλον, τόσο περισσότερο

μειώνονται οι πιθανότητες λήψης άχρηστων μπλοκ καθώς δε θα έχει περάσει αρκετός χρόνος από τη στιγμή που ενημέρωσε με το INFO μήνυμα.

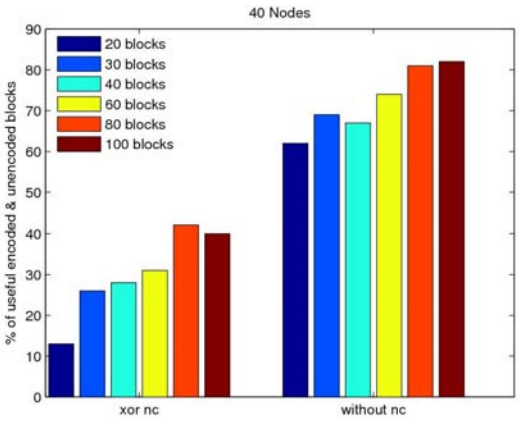
Στα ακόλουθα διαγράμματα φαίνονται τα ποσοστά των χρήσιμων μπλοκ που λαμβάνονται από τους κόμβους, για τα κωδικοποιημένα μπλοκ στο σύστημα με την κωδικοποίηση και για τα μη κωδικοποιημένα στο σύστημα χωρίς κωδικοποίηση για προσομοιώσεις με 1Mbps (αριστερή στήλη) και 10 Mbps (δεξιά στήλη). Τα αποτελέσματα είναι ενδεικτικά για πειραματικές προσομοιώσεις με 20, 40 και 80 κόμβους για τις οποίες παραπάνω παρουσιάζονται οι μέσοι χρόνοι ολοκλήρωσης του αρχείου και οι μέσες τιμές λαμβανόμενων Bytes/sec:



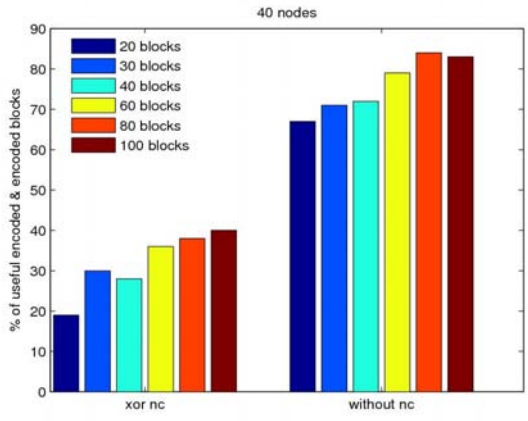
Σχήμα 31: 1 Mbps



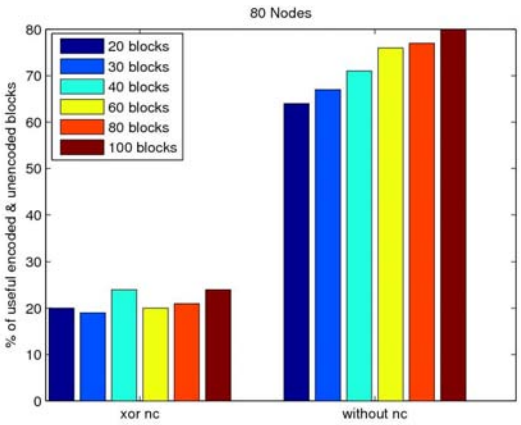
Σχήμα 32: 10 Mbps



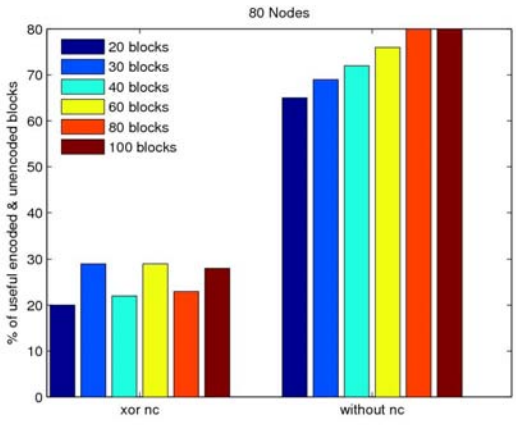
Σχήμα 33: 1 Mbps



Σχήμα 34: 10 Mbps



Σχήμα 35: 1 Mbps



Σχήμα 36: 10 Mbps

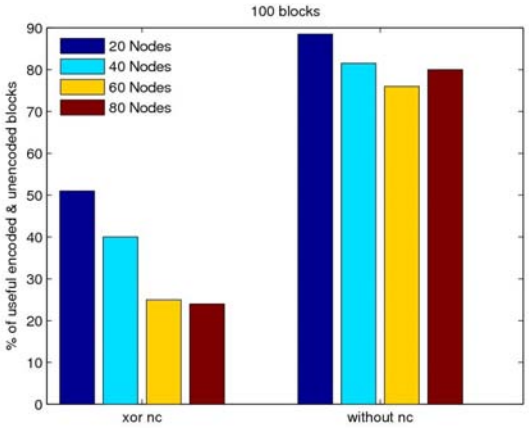
Όσο λιγότεροι οι κόμβοι στη γειτονιά τόσο περισσότερο στοχευμένες είναι οι αποστολές για τη διαδικασία της κωδικοποίησης, καθώς η πολυπλοκότητα είναι μικρότερη, όπως και οι καθυστερήσεις αποστολής πολλαπλών μηνυμάτων λόγω της επιλογής μικρότερου πλήθους κόμβων για πολλαπλή αποστολή.

Επίσης, όσο μεγαλώνει το πλήθος των μπλοκ, τόσο πιθανότερο είναι οι κόμβοι να έχουν μεταξύ τους μεγάλες αποκλίσεις στο ποια μπλοκ έχουν στην αποθήκη τους και έτσι οι πιθανότητες για ίδιες αποστολές και περισσότερα άχρηστα πακέτα μειώνονται. Αυτό φαίνεται περισσότερο στις προσομοιώσεις με λίγους κόμβους καθώς το πλήθος των κόμβων που θα σταλθεί το κωδικοποιημένο μήνυμα είναι μικρότερος.

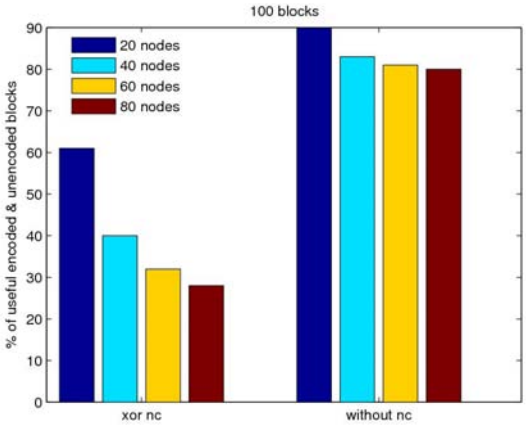
Όσο μεγαλύτερο δυνατό ρυθμό λήψης και αποστολής δεδομένων έχουμε, τόσο λιγότερο καθυστερεί η διαδικασία πολλαπλών αποστολών και το πλήθος άχρηστων μπλοκ μειώνεται καθώς δεν περνάει αρκετός χρόνος από τη στιγμή που έστειλε μήνυμα ενημέρωσης ο κόμβος για το ποια μπλοκ του αρχείου κατέχει.

Κατά ένα μέρος τα ποσοστά χρήσιμων και άχρηστων οφείλονται στην τυχαιότητα της επιλογής διαφορετικών μπλοκ από τους κόμβους.

Ακολουθεί ένα βοηθητικό διάγραμμα χρήσιμων λαμβανόμενων μπλοκ για 1 και 10 Mbps και σταθερό πλήθος μπλοκ:



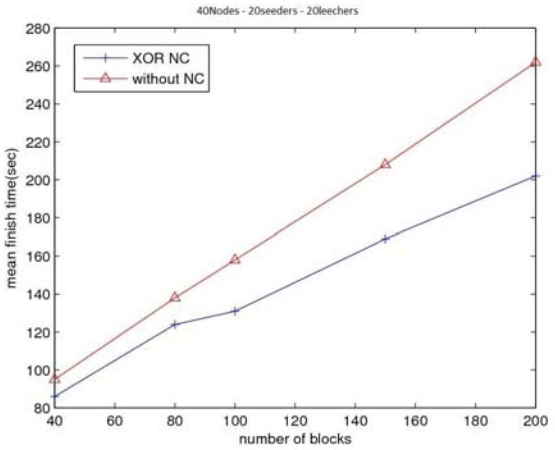
Σχήμα 37: 1 Mbps



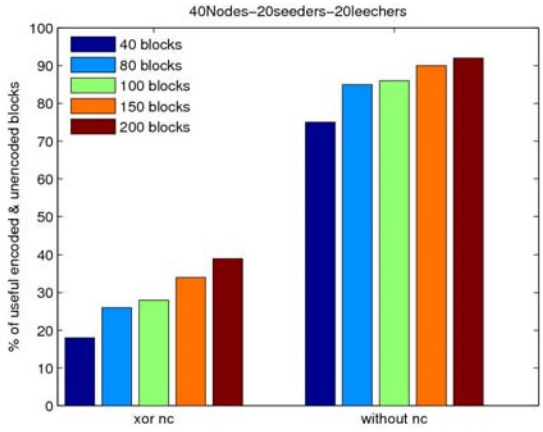
Σχήμα 38: 1 Mbps

Λόγος Seeders/Leechers

Επίσης, έγιναν πειράματα για διαφορετικές αναλογίες seeders – leechers στο δίκτυο κόμβων. Για 40 κόμβους, 20 από τους οποίους ήταν seeders και 20 leechers με 1Mbps ρυθμό αποστολής - λήψης, ο μέσος χρόνος ολοκλήρωσης των δύο συστημάτων καθώς και τα ποσοστά χρήσιμων μπλοκ φαίνεται στο παρακάτω σχήμα.

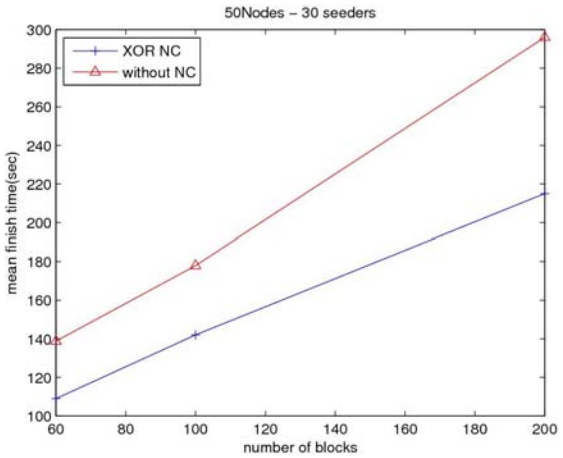


Σχήμα 39: 1 Mbps

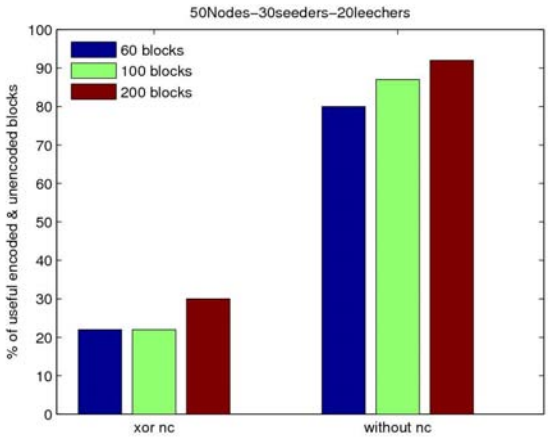


Σχήμα 40: 1 Mbps

Σε μία επόμενη προσομοίωση που έγινε θεωρήσαμε 50 κόμβους και 10 Mbps το bandwitdh, αλλά αυτή τη φορά με αναλογία 30 seeders και 20 leechers. Συνεπώς οι κόμβοι που έχουν το αρχείο στο δίκτυο προσομοίωσης είναι περισσότεροι από αυτούς που δεν έχουν το αρχείο. Ο μέσος χρόνος ολοκλήρωσης των δύο συστημάτων καθώς και τα ποσοστά χρήσιμων – άχρηστων μπλοκ φαίνεται στο παρακάτω σχήμα.



Σχήμα 41: 10 Mbps



Σχήμα 42: 10 Mbps

Επιλογή Τιμής Μετρικής

Όπως έχει αναφερθεί και προηγούμενα, η τιμή της μετρικής ορίζει το ελάχιστο ποσοστό του πλήθους των κόμβων, στους οποίους στέλνεται το κωδικοποιημένο μήνυμα, οι οποίοι θα μπορούν να αποκωδικοποιήσουν το μήνυμα. Ενώ το υπόλοιπο ποσοστό των κόμβων να το αποθηκεύσει στην αποθήκη του με τα κωδικοποιημένα μηνύματα για μελλοντική αποκωδικοποίηση.

Εξετάστηκαν διάφορες τιμές για την μετρική όπως 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8 μέσω πειραματικών διαδικασιών για διάφορους συνδυασμούς πλήθος κόμβων- πλήθος μπλοκ. Η χρήση υψηλής τιμής της μετρικής για παράδειγμα 0.7, 0.8 είχε ως αποτέλεσμα τη μη εκπλήρωση της πολλές φορές, με αποτέλεσμα να μην στέλνεται ο συνδυασμός των πακέτων και να μην αξιοποιείται τελικά η πληροφορία που στάλθηκε από όλους τους προς όλους τους κόμβους. Επίσης, για τιμή 0.3 σε ορισμένα πειράματα παρα η καλύτερη επίδοση ενώ σε ίσο αριθμό πειραμάτων υπήρχε και η χειρότερη επίδοση όσο αφορά τον χρόνο ολοκλήρωσης.

Καταλήξαμε στην επιλογή της μετρικής σε 0.5, με κριτήριο τον γρηγορότερο μέσο χρόνο ολοκλήρωσης του αρχείου από τους κόμβους. Για την τιμή αυτή οι επιδόσεις ήταν σταθερές στις αρχικές προς τις μεσαίες θέσεις όσο αφορά το μέσο χρόνο ολοκλήρωσης σε σχέση με τις άλλες τιμές της μετρικής.

Συμπεράσματα – Μελλοντική Εργασία

Στην παρούσα εργασία μελετήθηκε η συμβολή της τεχνικής *γραμμικής xor - κωδικοποίησης μπλοκ* αρχείου σε δίκτυα ομότιμων κόμβων η οποία συνδυάζεται με την συνεχή παρατήρηση των περιεχομένων της γειτονιάς ενός κόμβου. Στο παρόν μοντέλο η κωδικοποίηση αποτελεί μία υβριδική λειτουργία, με την έννοια ότι δε βασίζεται εξ ολοκλήρου στην κωδικοποίηση δικτύου για όλα τα μπλοκ αλλά συνυπάρχει με τον μη κωδικοποιημένο τρόπο διαμοιρασμού.

Ύστερα από πειραματικές προσομοιώσεις μεταξύ του συστήματος που χρησιμοποιεί κωδικοποίηση και του συστήματος που δε χρησιμοποιεί κωδικοποίηση, παρατηρήθηκαν βελτιώσεις στο μέσο χρόνο ολοκλήρωσης του αρχείου για το σύστημα με κωδικοποίηση σε σχέση με το σύστημα χωρίς κωδικοποίηση. Οι βελτιώσεις αυτές σημειώθηκαν ιδιαίτερα όταν οι συνδέσεις των κόμβων μπορούν να υποστηρίξουν υψηλό ρυθμό αποστολής και λήψης δεδομένων της τάξης των Mbps.

Το ποσοστό των χρήσιμων μηνυμάτων είναι ένας παράγοντας που έχει περιθώρια βελτιστοποίησης, με στόχο την απελευθέρωση του δικτύου από το φόρτο που προκαλεί η πληροφορία, η οποία αποβαίνει άχρηστη για τον κόμβο που τη λαμβάνει. Έτσι, το σύστημα θα χρειάζεται λιγότερους πόρους και θα έχει λιγότερες καθυστερήσεις αφού δε θα στέλνει άχρηστα μπλοκ. Επίσης, θα είναι περισσότερο ανθεκτικό για χαμηλότερους ρυθμούς αποστολής και λήψης πληροφορίας καθώς και πιο αποδοτικό σε δίκτυα με μεγάλο πλήθος κόμβων. Κάτι το οποίο θα μπορούσε να μελετηθεί ως προς αυτή τη κατεύθυνση των ποσοστών χρήσιμων μπλοκ στο σύστημα της κωδικοποίησης είναι και η τιμή της μετρικής. Εκτός από το κριτήριο του μέσου χρόνου ολοκλήρωσης για την επιλογή της μετρικής, θα ήταν επίσης ενδιαφέρον να συνυπολογιστεί και το κριτήριο των innovative μπλοκ για την επιλογή της τιμής της. Αυτό θα μπορούσε να κάνει το σύστημα πιο ευπροσάρμοστο σε δίκτυα με σχετικά χαμηλό ρυθμό αποστολής και λήψης πληροφορίας.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] http://xbtt.sourceforge.net/udp_tracker_protocol.html

[2] http://bittorrent.org/beps/bep_0005.html

[3] C. Gkantsidis and P. Rodriguez. "*Network coding for large scale content distribution*". In IEEE Incofom,Miami, FL, 2005.

[4] Christos Gkantsidis, John Miller, Pablo Rodriguez. *"Anatomy of a P2P Content Distribution system with Network Coding"*.

[5] Ingmar Baumgart, Bernhard Heep, Stephan Krause, "OverSim: *A Flexible Overlay Network Simulation Framework"*.

[6]Bram Cohen.*"Incentives Build Robustness in BitTorrent"*

[7]Raymond W. Yeung. *Avalanche: "A Network Coding Analysis"*

[8] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. "*Network information flow*". IEEE Transactions on Information Theory, July 2000

[9] Christina Fragouli, Jean-Yves Le Boudec, Jorg Widmer: "*Network Coding: An Instant Primer"*

[11] Philip A. Chou , Yunnan Wu , Kamal Jain. "*Practical Network Coding (2003)"*

[12] B. Gajic, J. Riihijärvi, and P. Mähönen, "*Performance Evaluation of Network Coding: Effects of Topology and Network Traffic for Linear and XOR Coding*"

[13] http://en.wikipedia.org/wiki/Peer-to-peer

[14] http://en.wikipedia.org/wiki/Network_coding

[15] http://www.omnetpp.org/doc/omnetpp41/manual/usman.html

[16] http://p2pfoundation.net/The_Foundation_for_P2P_Alternatives